2016-04-28

Perception Framework for Activities of Daily Living Manipulation Tasks

Koushik Balasubramanian
Worcester Polytechnic Institute

Perception Framework for Activities of Daily Living Manipulation tasks

by

Koushik Balasubramanian

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Robotics Engineering

by

May 2016

APPROVED:

Professor Taskin Padir, Thesis Advisor

Professor Michael Gennert

Professor Jie Fu

Abstract

There is an increasing concern in tackling the problems faced by the elderly community and physically in-locked people to lead an independent life experience problems with self-care. The need for developing service robots that can help people with mobility impairments is hence very essential. Developing a control framework for shared human-robot autonomy will allow locked-in individuals to perform the Activities of Daily Living (ADL) in a flexible way. The relevant ADL scenarios were identified as handling objects, self-feeding, and opening doors for indoor navigation assistance. Multiple experiments were conducted, which demonstrates that the robot executes these daily living tasks reliably without requiring adjustment to the environment. The indoor manipulation tasks hold the challenge of dealing with a wide range of unknown objects. This thesis presents a framework developed for grasping without requiring a priori knowledge of the objects being manipulated. A successful manipulation task requires the combination of aspects such as environment modeling, object detection with pose estimation, grasp planning, motion planning followed by an efficient grasp execution, which is validated by a 6+2 Degree of Freedom robotic manipulator.

Acknowledgements

I would like to express my gratitude to my advisor, Professor Taskin Padir, for giving me the opportunity to work on the project mentioned in this thesis and his invaluable support throughout the duration of this work. I would also like to thank Professor Michael Gennert and Professor Jie Fu for being a part of my advising committee and providing insights into different aspects of the research presented. Worcester Polytechnic Institute(WPI) has been a great place to learn and Robotics and Intelligent Vehicles Research (RIVeR) laboratory has been a great place to collaborate and work. Thanks to Vinayak Jagtap for his invaluable lessons on programming concepts that made me a good software developer. Thanks to Dmitry Sinyukov for his patience during my initial learning phase with JACO manipulator, thanks to Nandan Banerjee for helping me in bug fixes and thanks to Xianchao Long for his support in setting up the TrajOpt. I would also like to thank my dearest friend Satish Kumar Iyemperumal for his encouragement and support throughout the work. Lastly, I would like to thank my loving parents Mr. Balasubrmanian and Mrs. Nirmala, my elder brother, Karthik Balasubramanian, and other friends for their motivation and support.

Contents

List of Figures

List of Tables

Abbreviations

ADL	**A**ctivities of **D**aily **L**iving
AAL	**A**mbient **A**ssisted **L**iving
PCL	**P**oint **C**loud **L**ibrary
ROS	**R**obot **O**perating **S**ystem
MOPED	**M**ultiple **O**bject **P**ose **E**stimation and **D**etection
PCA	**P**rinciple **C**omponent **A**nalysis
TrajOpt	**Traj**ectory **Opt**imizer

Symbols

$\mathcal{P}$	Point Cloud Structure
$p_i \in \mathcal{P}$	A 3D point $\{x_i, y_i, z_i\}$ in a point cloud data
$\mathcal{P}^k$	points located in the k-neighborhood of a point p_i
$\overrightarrow{n}_i$	a surface normal at point p_i
$\mathcal{T}$	Transformation matrix containing Position and Orientation of the object
$\mathcal{C}$	Centroid of a set of Points $\mathcal{P}$
$\mathcal{Q}$	Configuration space of the manipulator
$q_t \in \mathcal{Q}$	Configuration of the robot at time t

Chapter 1

INTRODUCTION

As autonomous operations with Robots are continuously evolving, the ability to achieve a tight coupling between perception and action are starting to explore a wide range of applications. This research strives to push the intelligent manipulation one step towards achieving complete autonomy and as a result, I present a reliable perception framework for grasping and manipulation of household objects with and without prior knowledge of the object.

Service Robots is a promising research field and the International Federation of Robots (IFR) defines service robots as follows: A service robot is a robot which operates semi- or fully autonomously to perform services useful to the well-being of humans and equipment, excluding manufacturing operation [1]. According to IFR, about 4.7 million service robots for personal and domestic use were sold in 2014 and a total of 172,000 service robots for professional use have been sold since 1998 [2]. This engenders the need of sophisticated perception and manipulation planning algorithms such that the robot can robustly and reliably accomplish the household tasks.

1.1 Motivation and Goal

There are approximately 3.1 million people worldwide who are physically in-locked, experience problems with self-care due to the inability to interact flexibly with the physical world. People with motor and/or communication disabilities would like to interact with the environment and move to places independently. The primary motivation of this research is to serve these community of people to effectively perform the Activities of Daily Living (ADL) manipulation tasks. Human in the Loop Cyber Physical System

[1] http://www.ifr.org/service-robots/
[2] http://www.ifr.org/service-robots/statistics/

(HiLCPS) [1] deals with the challenge of restoring the autonomous actions by allowing the users to communicate and control devices in physical world through an embedded system. Robot Assistive Technology, a primary focus of HiLCPS, is my inspiration to work on this problem.

The goal of this research is to develop sophisticated perception algorithms to reliably detect and estimate pose of unknown objects and model the environment for grasping and manipulating household objects. The motion planner used for manipulation task is Trajectory Optimizer (TrajOpt) [2], a sequential convex optimizer planner, with a determined set of constraints. Developing a generalized constrained motion planner is out of the scope of this project.

1.2 Ambient Assisted Living (AAL)

In The United States of America, people over 65 years are the group of the population that is fastest growing in numbers, and by 2020 they are expected to represent 1 in 6 of the population [3]. Due to their high dependency ratio [4], there is a desire to assist the living at home with smart technologies. A potential solution to this problem is by extending the time people can live in their preferred environment by increasing their autonomy, self-confident and mobility, and providing such solutions are developed under the term Ambient Assisted Living (AAL) [3].

1.2.1 Need for AAL

The main aim of Ambient Assisted Living (AAL) is to improve the life quality of elderly people who need special care and assistance by providing cognitive and physical support and access to the environment services.

The *International Classification of Functioning, Disability, and Health* (ICF) attempts to bridge many of these definitions by considering disability as an umbrella term for impairments, activity limitations, and participation restrictions. The United States of America census [4] says that around 18.7 % of total US population possess some kind of disability, and 12 %, with ± 0.2 margin of error, suffer from sever disability which includes severe impairments and activity limitation. Figure 1.1 shows the distribution of category of people with disabilities and severe disabilities amongst the total US population of 291 million in 2010 released by US census in 2012.

[3] AAL was initially termed in Europe
[4] http://www.census.gov

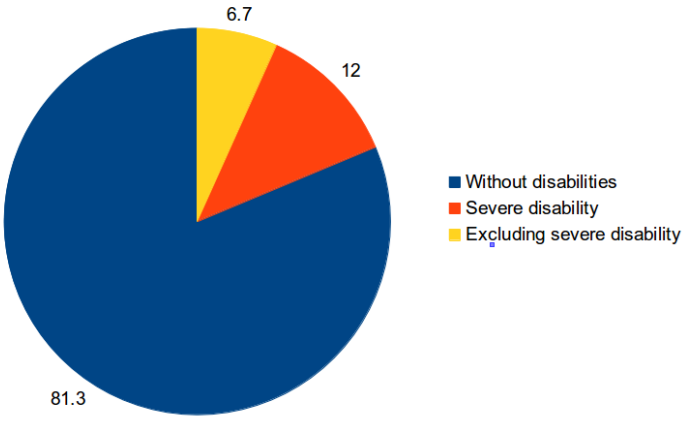


FIGURE 1.1: Prevalence of disability (in Percentage of population) in United States in 2010

The high disability ratio among the population of US will have some adverse consequences including, but not limited to, rising health care cost, an increase in age-related diseases, shortage of professionals / care-takers, an increase in the number of individuals unable to live independently. This explicitly creates an alarming desire to develop living assistance systems. Hence, the assistive devices are developed to facilitate the daily lives of these elderly people. These technologies would promise to help the elderly people living independently in comfortable ways.

1.2.2 Robotics in AAL

Assistive robots provide the major tool and infrastructure towards AAL as the robots are not only helpful in the physical tasks, but also serve as a communicator as people consider them as social entities. There are several assistive robots developed to reduce the need of movement of the user like vacuum cleaning robot, pick and place robot, nursing care robot like RIBA [5]. DUSTY [6] is a joy-stick controlled assistive robot that uses a manipulator to pick any object in the ground weighing less than a pound. Figure 1.2 shows the compact mobile robot DUSTY with a vertical lift to pick up and deliver dropped objects to people with motor impairments.

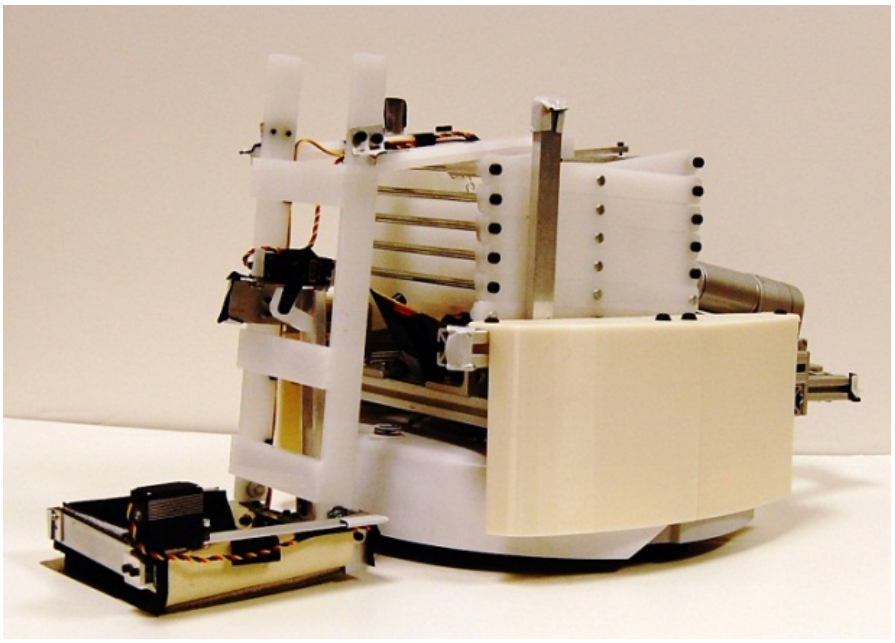


FIGURE 1.2: Pick up and deliver mobile robot DUSTY

1.3 The Robotic System

Mobile manipulators are a category of robotic systems in which a mobile platform is combined with a robotic manipulator. The mobile manipulator referred throughout this thesis is Anna [7], shown in Figure 1.3, is a semi-autonomous wheelchair combined with a 6 Degrees of Freedom manipulator. The mobile platform is a 2 Degrees of Freedom (DoF) with a translation and a rotational component. This platform is equipped with several sensors including ultrasonic sensors for collision avoidance, LiDAR for scanner data, encoders for odometry data and a Primesense ASUS camera for 3D point cloud data [8] [5]. The manipulator is a 6 DoF arm JACO from Kinova Robotics [6]. The manipulator is equipped with position sensor at each joint for state estimation and an end-effector force/torque sensor.

Integrating the mobile platform with a manipulator results in a system with redundant kinematics as the mobile manipulator possesses more than 6 DoF with the task space being only 6 dimensional. Solving redundant kinematics of the mobile manipulator is out of the scope of this thesis as I enforce the mobile platform to be in rest while the manipulator is active. In all the test cases considered during validation, the objects being manipulated are in a small region within the reach of manipulators work space.

[5] Point Cloud Library presents an advanced approach to 3D perception with state of the art algorithms for processing the data

[6] www.kinovarobotics.com/

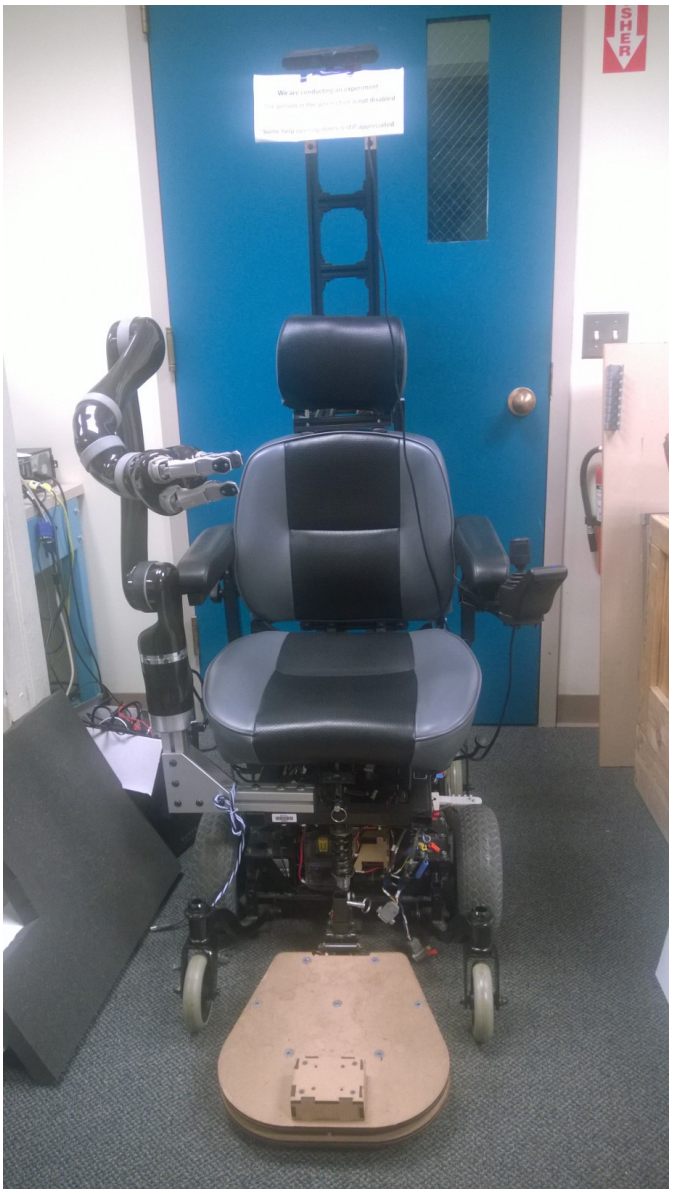


FIGURE 1.3: The mobile manipulator Anna used in this thesis

1.4 Activities of Daily Living Tasks

Census reports that in the US, about 12.3 million people aged 6 years and older, constituting 4.4 % of population, needed assistance with one or more Activities of Daily Living (ADLs), Enhanced Activities of Daily Living (EADLs) and Instrumental Activities of Daily Living (IADLs). The ADLs include difficulty getting around inside the home, getting into/out of bed, bathing, dress-ing, eating, use transportation or toileting. The EADLs include participation in social and enriching activities, such as learning new skills and engaging in hobbies. The IADLs include difficulty going outside the home, managing money, preparing meals, doing housework, taking prescription medication, and using the phone. IADLs are generally more cognitive demanding than ADLs. These

categories constitute most of the tasks older adults spend their time performing in the home environment; essentially, older adults want to make their time there as enjoyable and productive as possible [9].

TABLE 1.1: Subset of categories of ADL and approximate number of people (in thousands) in need of assistance

Category of ADL	Number(in US)	Needed assistance	Percentage %
Difficulty getting around	4,552	2,452	53.86
Difficulty eating	1,845	1,031	55.8
Difficulty doing housework	7,708	5,892	76.44

Some of the relevant ADL scenarios identified in this project were door opening for navigation assistance, self feeding amd identifying and manipulating household objects that are commonly used. Table 1.1 shows the approximate number of people (in thousands) who definitely need assistance for these above identified ADLs among the US population in 2010 released by US government census data is 2012.

1.4.1 Challenges involved in ADL manipulation tasks

ADL manipulation tasks involves several key design requirements include safety, dependability, modularity, reliability and fault handling in the system. The robot will be closely working with human and the human-robot shared autonomy tasks require high degree of safety measures including collision checks, limited workspace, efficient and reliable environment modeling. Performing everyday manipulation tasks involves computing motions that are subject to multiple task space constraints and these are to be addressed in high dimensional C-space manipulation planning.

The most common challenge in performing ADL manipulation task is the constraints on the pose of the robot's end-effector. For example, carrying a cup of water to the user possess the constraint of no rotation in the end-effector. The 3D rotational component of the end effector, described by the Special Orthogonal group $SO(3)$, is fixed throughout the entire path, so as to not to spill the water. Reaching the manipulator to grasp an object also imposes constraints on grasp configuration based on the object's 6D pose and this constraint is satisfied by efficiently computing the approach vector of end effector for grasp planning.

The other computational problem arising from the perception of the environment from the 3D point cloud data to differentiate between the objects and planar surfaces like tables and cupboards for efficient segmentation will be discussed in the later chapters.

1.5 Overview

The graphical roadmap illustrating the organization of the thesis is shown in Figure 1.4. This thesis describes various Activities of Daily Living tasks performed using the wheelchair - manipulator robotic system Anna. Chapter 2 deals with a perception technique, Multiple Object Pose Estimation and Detection (MOPED), that has the capability of detecting multiple objects in a cluttered scene as well as estimating their 6D pose in the scene. Chapter 3 demonstrates the primary ADL task of the robot assisting in opening the door for indoor navigation and describes the entire perception framwework for handle detection and grasp estimation. Chapter 4 demonstrates another ADL task of robot assisting the user in self feeding and describes the task space constraints involved in the task. Chapter 5 describes the results obtained from the capabilities of robot assisting the user in ADL task and Chapter 6 concludes with the discussion and future work to extend so as to achieve complete autonomy in manipulation.

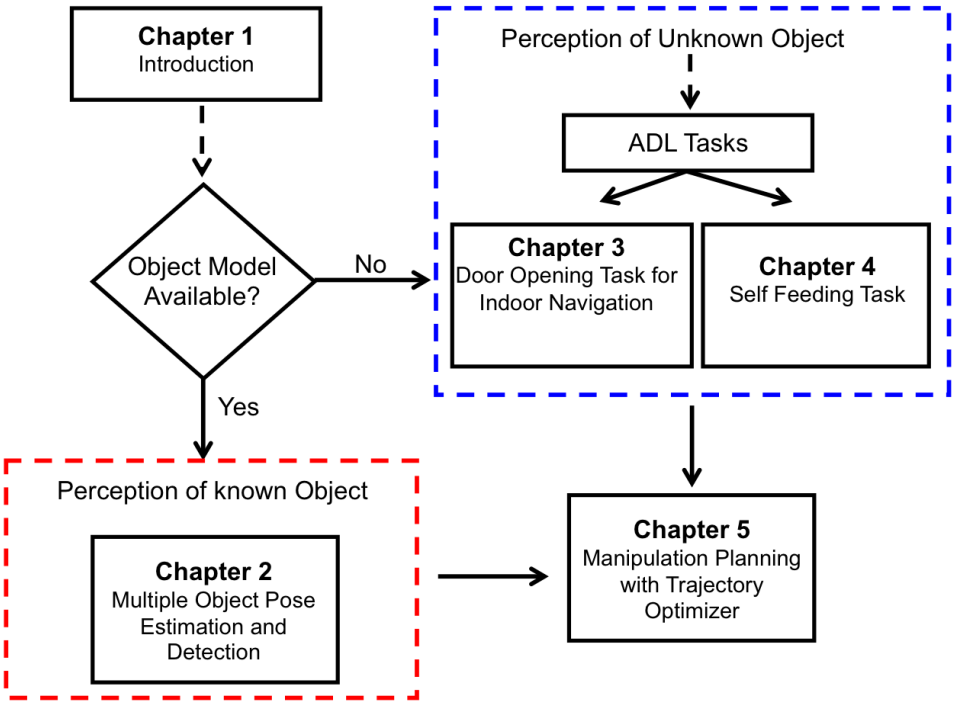


FIGURE 1.4: Graphical roadmap of the thesis

Chapter 2

MULTIPLE OBJECT POSE ESTIMATION AND DETECTION

Object detection is one of the most fundamental and at the same time one of the most complex problem in the field of computer vision. Multiple Object Pose Estimation and Detection (MOPED)[10] framework is a multi-image object recognition and pose estimation technique that uses a novel pose clustering technique for handling outliers. A part of this thesis focuses on implementing the MOPED, generating models of new objects, detecting the pose of those objects in the scene and detecting the modelled objects with their pose in a scene. This project was started as a part of course project in a team of 3 and I extended the work to obtain the 6D pose and estimate the grasp pose for JACO manipulator.

MOPED uses Scale Invariant Feature Transform(SIFT) to generate Keypoints and descriptors of an image. These Keypoints are then matched with scene to determine if an object is present. SIFT is a robust technique of detecting objects as it is invariant to scale, rotation(upto 60 degrees), translation, partially invariant to illumination changes. MOPED not only gives position of the objects but also provides 3D pose information. It is one of the most effective framework to detect the object in the cluttered environment. Since MOPED also estimates 6D pose of the object, it is easier to integrate this framework with manipulation as this 6D pose will be the goal pose of the end-effector of manipulator for grasping tasks.

2.1 Modeling the object

MOPED requires a 3D model of the object to be detected which is generated using Structure From Motion (SFM), that extracts geometric structure of the object from images through a camera's motion, using functions within OpenCV's API.

2.1.1 Structure From Motion

MOPED requires at least 24 images of the object spanning 360°, where any point on the object should be visible in at least 3 images. But not every image is ideal. To account for bad images, 50-60 images were taken. Object should be kept on sheet of paper with lots of high frequency features. Figure 2.1 shows a sample set of images given as input for model creation where the soap bar is captured from different angles as compared to the laptop which serves as a reference feature for rotation and scaling.

FIGURE 2.1: Generating the 3D Model of the object by Structure From Motion Technique

Using SFM principles, a 3D point cloud of Keypoints can be generated using a python tool *model_generator*. This tool finds SIFT features in all the available images of object. The Keypoints that are generated are passed to Bundler[11] along with the camera's intrinsic parameters to generate a sparse 3D point cloud data of the object in the scene.

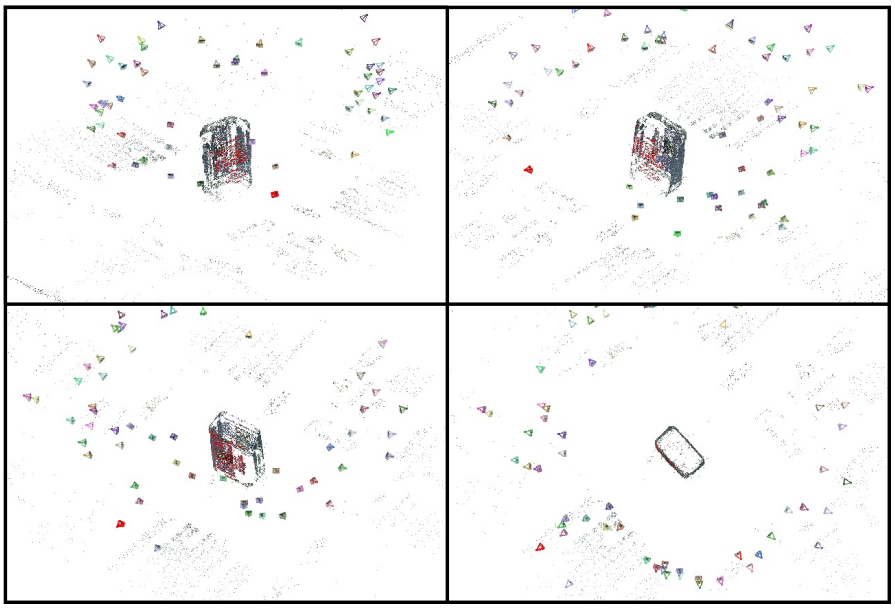


FIGURE 2.2: 3D Model of soap created using Bundler

SIFT features of these input images are matched to each other. Based on the matching, different camera poses are calculated and the position of images is determined. Keypoints belonging to these images are clustered together in the model. Since surroundings of the object are intentionally kept cluttered to get the orientation, these environmental elements are visible in the 3D model as well. Figure 2.2 displays 3D model of the soap from different angles.

Knowledge of camera's intrinsic parameters is crucial for generating accurate model. These parameters can be extracted from Exchangable Information File format (EXIF) tags of images, however calibrating the camera and providing these parameters in XML file improves the quality of model. For details on camera calibration, see Appendix B.

2.1.2 Refining the 3D Model

Once a 3D point cloud is generated, masks are applied to every image to remove outliers, which are the environmental elements used for determining orientation. Mask is a binary image that has the object pixels as white (true) and remaining part of the image is black (false). Figure 2.3 displays an example image of soap on left and it's mask on right. Model is refined and the environmental elements are removed using these masks.

The process described above generates the model in XML format. This file contains all the Keypoints repeated for every camera pose. Descriptors for the Keypoints are also

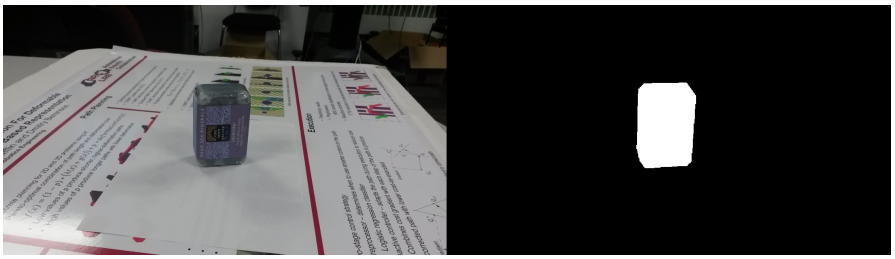


FIGURE 2.3: Soap image on the left, Mask of the image on the right

stored in the same XML tag. Path to this model is provided when running the MOPED node so that it can compare Keypoints in incoming images with the ones stored in the model file.

2.2 Methodology and Implementation

Moped uses SIFT features to obtain Keypoints and to extract descriptors from natural object features. The system is separated into an offline object modelling stage and an online recognition and registration stage. The invariant local features are used to obtain point matches between multiple 2D images of a rigid 3D object or scene. These are then used as input to bundle adjustment to obtain a metrically accurate 3D solution for the locations of the features and cameras. 3D models are used for recognition as they integrate features from various views and are therefore more complete and robust. The first stage, object modeling was discussed in the previous section.

The second stage of this process deals with the object recognition and 6 Dimensional solution of the model pose. Feature detected in the video frame are matched to those of the world model and the current pose of the model is estimated with these matches. The object detection and pose estimation process can be split into 7 tasks as described below:

2.2.1 Feature Extraction

For every input image, features are extracted using SIFT[12]. These are usually high frequency features called as Keypoints. SIFT uses an approximation of Laplacian of Gaussian(LoG), known as Difference of Gaussians(DoG) to speed up the process. This is done for different octaves of the image in Gaussian Pyramid as shown in Figure 2.4. Images are then searched for local extrema which are potential Keypoints. Edges are removed using Hessian Matrix to finalize the Keypoints. Once Keypoints are finalized,

Orientation is assigned to every keypoint to achieve rotation invariance. This information of a keypoint and it's neighbors is stored into a 128 dimension vector known as Descriptor. Information of neighbors in the descriptors allow detection of object with partial occlusions to a certain degree.

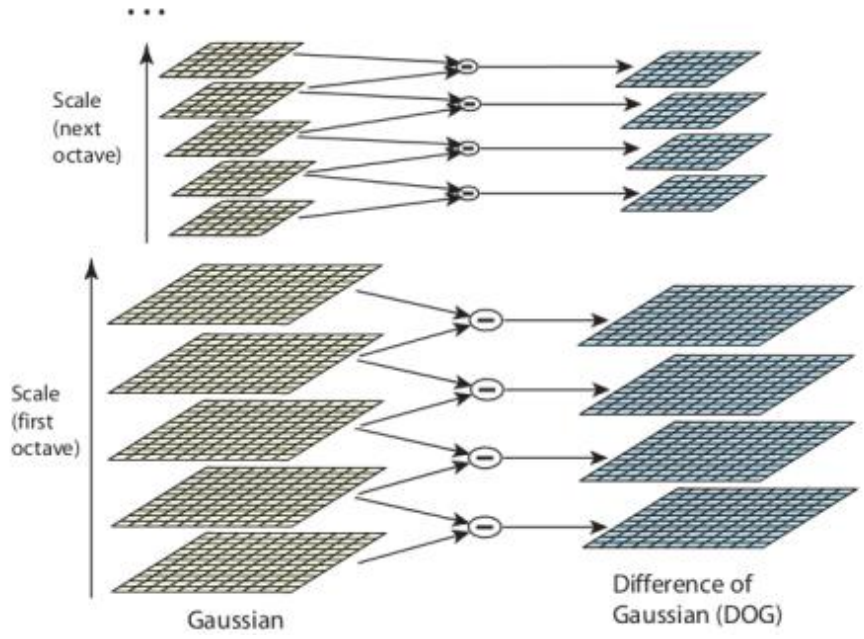


FIGURE 2.4: Gaussian Pyramid

2.2.2 Feature Matching

SIFT features are matched by a finding their approximate nearest neighbors, defined as the feature with the minimum Euclidean distance between descriptor vectors. But approximate matching technique may produce more outliers. As the exact 6 Dimensional pose of the object is required, the orientation of each of the descriptor must also be found. One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations. Fast Library for Approximate Nearest Neighbors (FLANN) [13] is used for matching descriptors.

2.2.3 Feature Clustering

Feature clustering is important because spatially close features are more likely to belong to the same object instance. Mean shift algorithm [14] is used to locate the maxima of the density function used for clustering. The features extracted from the image

are considered as an empirical probability distribution function. The dense regions corresponds to the local maxima of the probability distribution function. For each data point, Mean shift associates it with the nearby peak of the datasets probability density function. For each data point, Mean shift defines a window around it and computes the mean of the data point . Then it shifts the center of the window to the mean and repeats the algorithm till it converges. After each iteration, the window shifts to a denser region of the dataset.

2.2.4 Hypothesis Generation

After the clustering of features are done, we need to generate an hypothesis to which clusters of the image belong. Initially it is assumed that a cluster of feature belongs to a single image. Process each cluster independently in search of objects. RANdom SAmple Consensus (RANSAC), which finds the best fit line(s) through the feature clusters is used to find object instances that are loosely consistent with each objects geometry in spite of outliers. After this step, a rough pose of the object is obtained as I get the object descriptor along with it's orientation in the real space. Coarse object detection and pose estimation is done but with the outliers of the image included in the scene.

2.2.5 Pose Clustering

The clusters that are formed in the previous steps are again clustered to form a bigger cluster. As the same object might be present in multiple clusters, re-cluster image space features using poses resulting from Hypothesis Generation so as to avoid the redundancy. The same RANSAC method is used to obtain the new cluster. New, larger clusters are created, that often contain all consistent features for a whole object.

2.2.6 Hypothesis Refinement

After a coarse object detection is done, some refinement of the image space features is necessary for a fine object detection. After re-clustering and merging, most of the outliers are removed. Each of the new clusters contain features corresponding to only one instance of an object. Increase the iteration value of RANSAC to estimate a single pose from each cluster, but this has a trade-off with the computation time for the entire manipulation planning.

2.2.7 Pose Recombination

Similar poses that might be present by any chance are removed here by again merging together object instances with similar poses. This final merging step removes any multiple detection of the same object thus effectively forming a single pose for an object. A 6 Degrees of Freedom pose estimation is done by the above mentioned seven step algorithm of MOPED framework.

2.3 Limitation of MOPED in ADL scenarios

MOPED needs a pre-computed 3D model of the object to be detected and this limitation leads to the development of a generalized perception framework that can detect and estimate the pose and estimate the grasp configuration just from the 3D point cloud data and the object's surface geometry. Household objects can have multiple variation and a 3D Bundler model of all those variations is very tedious and its not a good approach. For example, in door opening task for autonomous indoor navigation, the robot has to identify the handle of the door, estimate its pose and manipulate it. MOPED needs the complete 3D model of this handle. But handle fitted in door is not a unique design and in order to address this, the following sections demonstrates a perception framework that can be implemented on objects with no prior knowledge.

Chapter 3

DOOR OPENING TASK FOR INDOOR NAVIGATION

A robust door handle detector is needed for the door opening task since the robot at home environment may have to function in a varying light conditions, varying door and handle specifications like handle types (lever and knob types), door and handle color for RGB segmentation and so on. To account for this, a 3D point cloud based approach of handle detection is developed which requires no prior model of the handle and can work on any lighting condition as this algorithm depend only on the 3D points.

3.1 Point Cloud Structure

A *point cloud* structure P is referred to a collection of 3 dimensional points in the free space. Point cloud data from a 3D Perception system like Primesense provides discrete and useful representation of the environment. A point in the 3D model is $p = \{x, y, z\}$ and the $\{x_i, y_i, z_i\}$ coordinates of any point $p_i \in \mathcal{P}$ are the distances from a given reference frame having its origin at the sensing device used to acquire the data. Each point p_i represents the distance on the 3D coordinate axes from the viewpoint of the sensing device. Distance of each point from the sensor is measured by the Time Of Flight (TOF) system, which measure the delay until an emitted signal hits a surface and returns to the receiver. Figure 3.1 shows the 3D Point Cloud data of a sample workspace obtained from Primesense. The data is represented by a format represented by the structure pcl::PointCloud$\langle pcl :: PointXYZRGB \rangle$, which represent the Euclidean XYZ coordinates and the RGB color of Each point in the space.

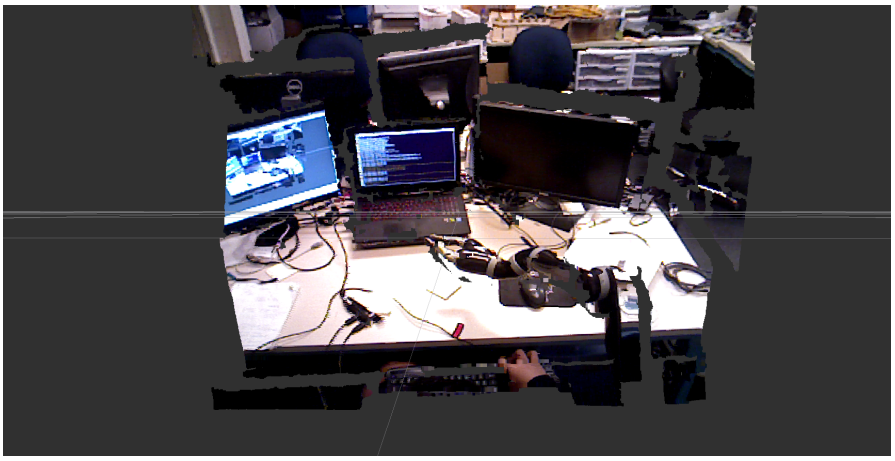


FIGURE 3.1: The obtained Point Cloud of the workspace displayed in RViz

3.1.1 Processing the acquired 3D Point Cloud data

Once a 3D point cloud data set has been acquired, several geometric processing steps has to be done such that meaningful information can be extracted to help the robot in performing tasks. The raw input point cloud data should be converted into different representation and formats.

Point cloud segmentation and point clustering are few of the notable processing steps that can work directly on the raw point cloud data. *Segmentation* is a process finding all the subset of points within the point cloud data, that supports an arbitrary model.The model can be a plane model, cylinder model and so on. Different types of segmentation process are listed in segmentation tutorial of PCL documentation [1]. For example, a mobile manipulation task of manipulating a household objects needs to extract out the objects separately by removing the point clouds of table plane. In our case of handle detection, the door plane, perpendicular to the ground needs to be removed by segmentation. The main purpose of *clustering* is to group similar structures together, in order to lower the computational resources needed by other subsequent algorithmic steps. Given an unorganized point cloud model $\mathcal{P}$, clustering divides the data into smaller parts so that the overall processing time for $\mathcal{P}$ is reduced significantly.

3.1.2 Downsampling the 3D Point Cloud data

Spacial decomposition techniques like kd-trees and octrees are commonly used techniques to partition a three dimensional space by recursively subdividing it. Similar techniques can be used in partitioning the point cloud data P into several chunks so as to make the search faster. Downsampling is a technique to reduce the number of points p_i from P

[1] http://pointclouds.org/documentation/tutorials/segmentation-tutorial

using a voxelized grid approach. The point cloud data P are converted into 3D Voxels, which are a set of tiny 3D boxes in space, of arbitrary width, length and height. Each 3D voxel, containing many point p_i, will be downsampled with their centroid. Higher the voxel volume, lesser the resultant number of points generated.

Downsampling the point cloud data not only provides fast access to the point location and feature searches, but also helps in environment modeling for collision avoidance application. Octree representation gives a better and faster estimate of distance from the spaces which are free or occluded.

3.2 Robot Operating System (ROS) architecture of the system

The entire wheelchair-manipulator system operates on ROS, hence the Point Cloud Library (PCL) perception framework developed in this project is also made ROS compatible. *pcl_ros* provides the necessary interface for running a ROS system to a PCL application. The approach in this project utilizes ROS for the entire communication pipeline, driver support and configuration needs. ROS makes this framework modular, with the intention of allowing to incorporate alternate motion planning, control or perception frameworks.

The architecture of the system is shown in Figure 3.2. The boxes with dashed line represents the group of nodes differentiated by their functionality. The boxes with continuous line represent a node. The arrow mark shows the communication between the nodes through ROS communication pipeline.

The main focus of this section is describing the perception framework developed for robust handle detection. Hence, this chapter covers the functioning of *Handle Detection*, *Pose Estimation* and *Grasp Estimation* nodes. *Grasp Execution* node is discussed in Chapter 5. The remaining components fall outside the scope of this thesis.

3.3 Handle Detection and Pose Estimation

The primary assumption of the handle detection task is that the wheelchair should have reached to a region proximity to the door, where the handle is within the workspace of the manipulator and the handle is in the field of view of primesense device. In order to reduce the computational and search time, the search space of the system is restricted to a part of the 3D space, which are in proximity of the door. Since this approach does

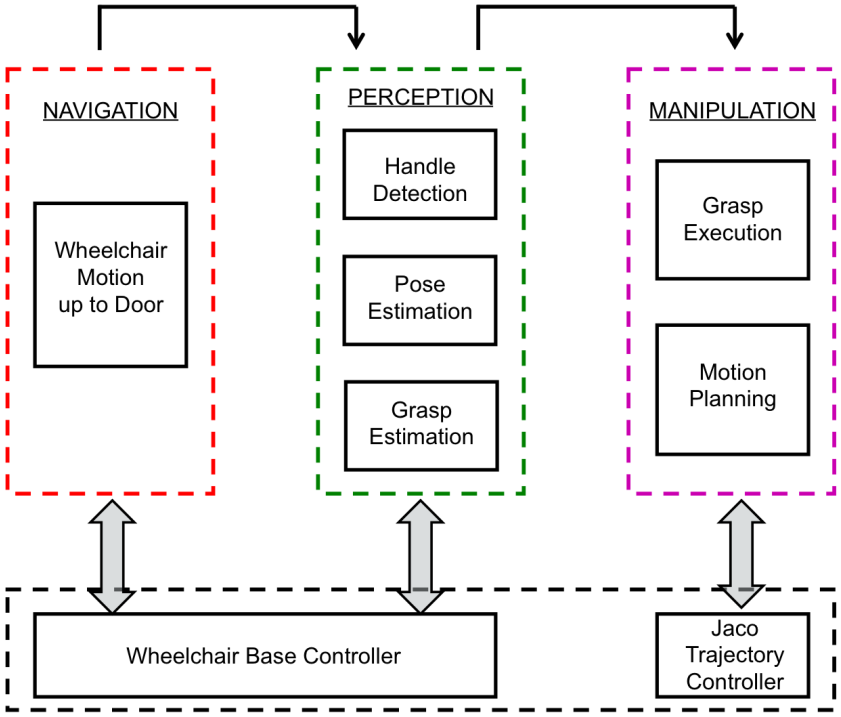


FIGURE 3.2: ROS architecture of the system representing the communication between the nodes

not rely on the 2D image geometry of the surface being detected, the handle detection method operates only on the Point Cloud data $\mathcal{P}$ discussed above.

3.3.1 Computational Problems

To demonstrate the algorithm, the lever type handle is considered. Handles possess very thin geometrical structure, and the 3D volume they occupy is very small, when compared to the volume of the door in the point cloud. Extracting a small subset of 3D points from a large 480 x 640 sized acquisition device is a complex task. Also, a bigger portion of the handle visible from the acquisition device as shown in the Figure 3.3, will have their surface parallel to the surface of the door. As a result, 2D plane based segmentation of Point Cloud data will fail because segmentation will remove the useful planar information of handle also.

Figure 3.4 shows another notable difficulty in detecting handles caused by the change in reflectivity of the handle surface in varying lighting condition. The intensity data is really useful while performing object detection tasks that involve metal surfaces, but in this case, the sampled data becomes too noisy and sparse when the pcl::PointCloud$\langle pcl :: PointXYZRGBI \rangle$ point data type is used.

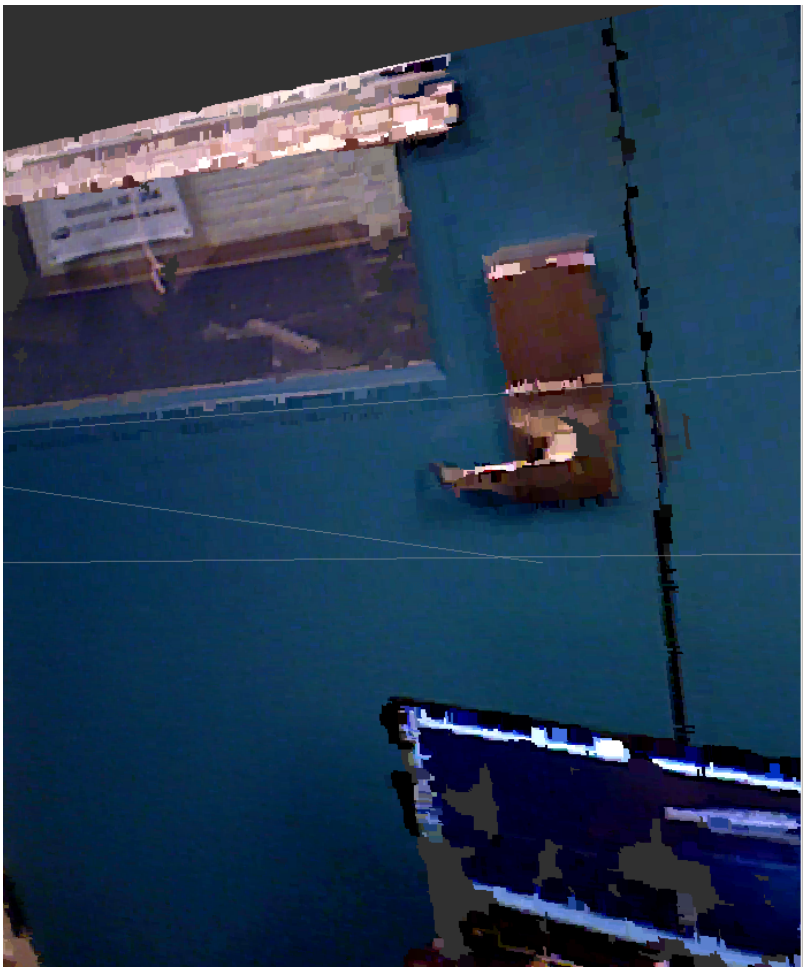


FIGURE 3.3: The Point Cloud data of the handle with the door surface

FIGURE 3.4: The change in reflectivity at varying lighting condition. Left: Environment with dull light Centre: Environment with bright light Right: A subset of door points having brighter intensity than the handle causing noise

Hence obtaining useful grasp information can be achieved only by performing complex

processing tasks like extracting the edges of entire point cloud by estimating the surface normal p_n of all the points $p_i = \{x_i, y_i, z_i\}$, extracting useful points related to the handle and then applying Principle Component Analysis (PCA) on the points to obtain the mean magnitude and direction of the aligned points. The further sections explains these steps in detail to achieve a robust handle pose estimation.

3.4 Normal Estimation and 3D Edge Detection

Surface reconstruction needs consistently oriented normals by assigning a vector $\overrightarrow{n}_i = \{n_x, n_y, n_z\}$ at each point p_i of the point cloud. A local plane is fitted to each point with their corresponding neighbors to estimate the direction of the gradient. The line normal to the local directional change is the normal to that point. Figure 3.5 shows a simple illustration of normal being computed on a circular surface.

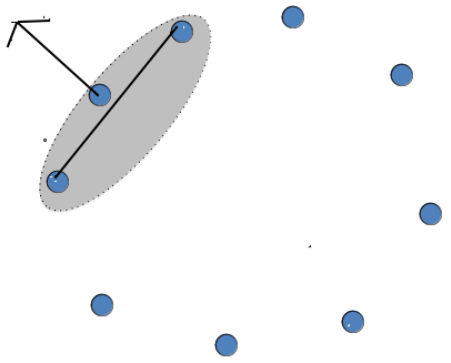


FIGURE 3.5: Normal of a point on a circular surface

This approach will be very expensive in huge 3D point cloud data as it is expensive to run a spanning tree. A simpler and efficient approach is to use local plane fitting that estimates a plane instead of a normal vector. For each point $x_i \in \mathcal{P}$, pick k nearest neighbors, $\{x_1, x_2, ...x_k\}$ and find a plane π that minimizes the sum of squared distances (SSD). For a set of k points, there may be many number of 2-dimensional plane available. From the set of planes, estimate the plane that has minimum SSD.

$$\min \sum_{i=1}^{k} dist(x_i, \pi)^2$$

In this approach, I used $k = 8$ with a simple logic that a point in the 2-dimensional plane will have 8 neighbors, with the general case being $3^n - 1$, where n is the dimension of the plane. The least square plane fitting estimation problem approximates the problem of determining the normal $\overrightarrow{n}_i = \{n_x, n_y, n_z\}$ to a point p_i on the surface to the problem of estimating the normal of a plane tangent to the surface that minimizes the SSD. A much more complicated and computationally expensive problem of estimating surface normal at all sample points of the point cloud based on local least square fitting is discussed in [15].

After the normals $\overrightarrow{n}_i$'s are estimated, 3D RGB-Depth edge detection is feasible. RGB-D edge detection based on the 3D geometric information and photometric information is demonstrated in [16]. The */depth/registered_points* ROS topic of Openni [2], the SDK containing binaries and drivers for primesense, contains the organised point cloud of the scene. The organized point cloud is similar to images with rows and columns of pixels containing the information of points in scene. As a result, neighbor search is done with the row and column indices instead of performing a time-consuming 3D search, as is necessary for general unorganized point clouds.

To determine the edges inside the point cloud, local 8-Neighbor search is performed. The point cloud is separated as an Octree representation, which is a variant of k-d trees that splits into successive branches of equal cubical volume. The section 3.4.1 describes more about the k-d tree implemented on the Point Cloud data along with the search technique. In a 3-dimensional Octree, the (x, y) component is considered along with the Depth information $\mathcal{D}$ at each point. The local 8-neighbor search is performed via 8-Neighbor$(\mathcal{D}, x, y)$. This enables the calculation of maximum depth difference from the current location to all the local neighbors. High curvature edges can also be efficiently detected in 3D points by using a variant of Canny edge detector [17]. The first order image gradients of Canny filters G_x and G_y are applied to the surface normal $\mathcal{N}$ of the image with the components of normal being N_x and N_y. Figure 3.6 shows the detection of high and low curvature edges on the door handle Point Cloud data.

3.4.1 k-d tree representation of a point cloud

K-d tree is a binary search tree with constraints imposed on the search direction. The K-d tree implementation in this approach deals with 3 dimensional point clouds, hence each level of a k-d tree splits all children along a specific dimension, using a hyperplane that is perpendicular to the corresponding axis. This is a kind of exhaustive search for

[2] http://structure.io/openni

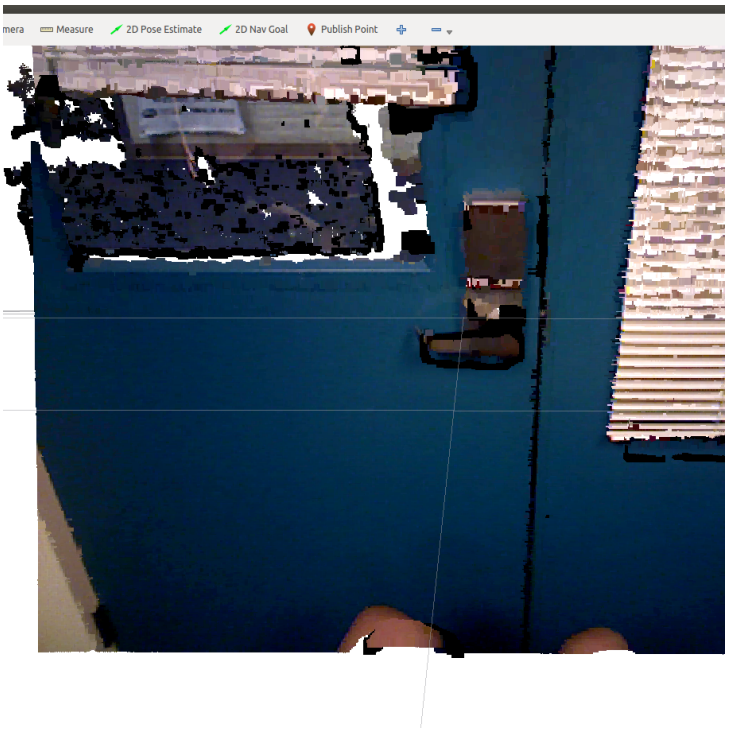


FIGURE 3.6: 3 dimensional edge detection result on the point cloud containing the handle

the worst case as the search should return to the first dimension of all others have been exhausted. Figure 3.7 shows an example partition method for a 2-dimensional k-d tree.

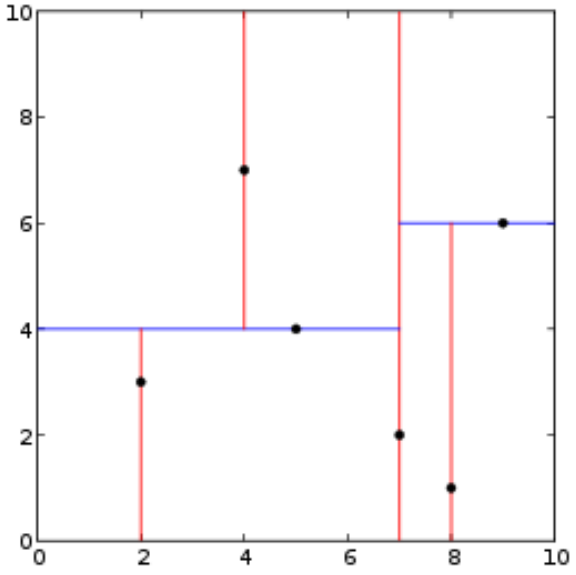


FIGURE 3.7: A 2d k-d tree from the k-d tree tutorial in PCL documentation

3.4.2 Clustering and estimating inliers

From the boundary information obtained as shown in Figure 3.8, handles need to be extracted out separately for Pose estimation. We known that the edges of the handles that are parallel to the ground plane are parallel to each other when viewed at an angle normal to the door plane, and almost parallel to each other when viewed from a different angle.

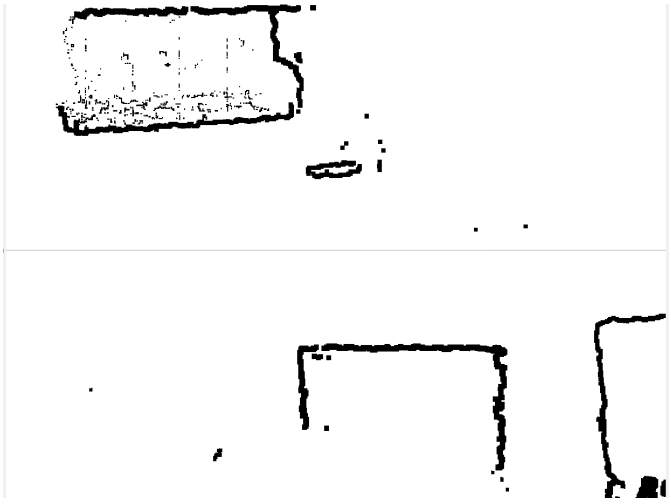


FIGURE 3.8: Extracted edge indices from the scene displayed in RViz

Another useful information that can be used to extract the handle indices alone is the distance of those points pcl::PointCloud$\langle pcl :: PointXYZ \rangle$ from the origin transformation frame, an arbitrary point in the ground plane, is known. Figure 3.9 shows the difference in color based on the distance d_i of the points $\{x_i, y_i, z_i\}$ from the origin.

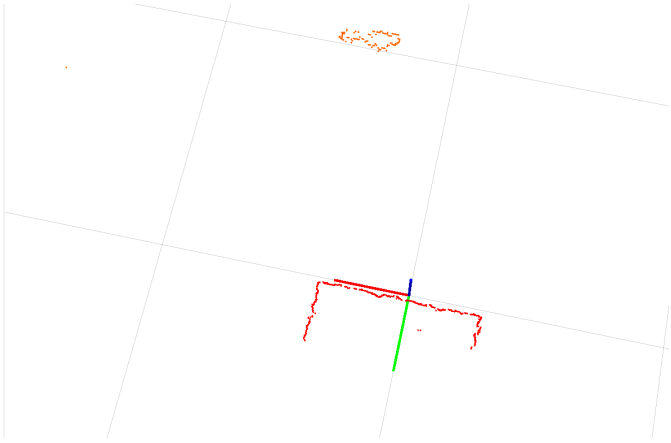


FIGURE 3.9: Observed changes (from top view) in color of points due to varying distance. The edges detected in the floor are marked in red and the edges detected in handle are marked orange

The approach used to extract the handle indices is straightforward and computationally easy. The indices containing the edge information are copied to a new point Cloud data. Probabilistic Hough Transform is applied over the edge image to extract only the lines. After checking the slopes of the lines, only vertical lines that have pixel length lesser that a predefined threshold (in pixels) are retained since the handle length is usually smaller than other lines arising through the door edges. All the vertical edges along with larger horizontal edges are eliminated. RANdom SAmple Consensus (RANSAC) algorithm [18]is a general parameter estimation approach. RANSAC calculates the line equation of the lines available and the two lines, that have similar slopes (with a ± 0.1 tolerance) separated by a small distance are extracted out as the handle indices. Figure 3.10 shows the extracted point cloud containing the handle data.

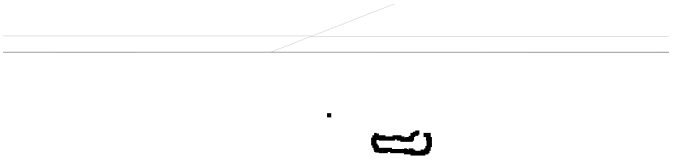


FIGURE 3.10: Extracted edge indices of the door handle displayed in RViz

The pose information containing the rotation of the handle with respect to the camera frame can be obtained by Principle Component Analysis and Section 3.5 discusses the algorithm in detail.

3.5 Principle Component Analysis (PCA)

The 3D edge detection gives out the boundary information of handle in terms of vector of pcl::PointCloud$\langle pcl :: PointXYZ \rangle$. Principle Component Analysis (PCA) [19] finds an orthogonal basis that best represents a set of observations. Given a set of Point cloud Points $\{x_1, x_2, ...x_n\} \in \mathcal{R}^3$, PCA finds the best approximating hyper plane passing through a point p with its normal $\overrightarrow{n}_p$ such that

$$min_{p,n} \sum_{i=1}^{n} ((x_i - p)^T n)^2$$

Satisfying the above equation will result in obtaining the Eigen vector $\overrightarrow{n}_p$ at p of the dataset, that determines the shift and correlation of the observation.

The centroid $\mathcal{C}$ $\{C_x, C_y, C_z\}$ of a dataset containing the input points $\{x_1, x_2, ...x_n\} \in \mathcal{R}^3$ is given by

$$\mathcal{C} = 1/n \sum_{i=1}^{n} x_i$$

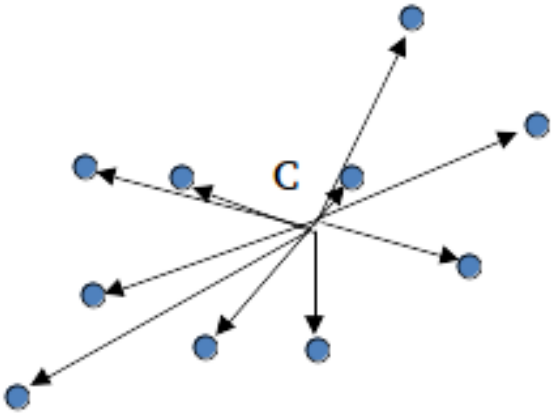


FIGURE 3.11: Vectors originating from centroid of the dataset in a 3-dimensional plane

Let the vectors from each point x_i to the centroid be y_i and represented by

$$y_i = x_i - \mathcal{C}$$

Figure 3.11 shows the vectors $\overrightarrow{y}_i$ from the centroid $\mathcal{C}$ to points $x_i \in \mathcal{R}^3$.

The Eigen vector of the Point Cloud points can be calculated along with a hyper plane that represents the homogeneous transformation matrix with the following steps:

- Compute the centroid $\mathcal{C}$ represented by $\{C_x, C_y, C_z\}$ and the normalized covariance of the input Point Cloud
- Compute the Eigen Vector for the 3-dimensional space $\overrightarrow{e}_x$, $\overrightarrow{e}_y$ and $\overrightarrow{e}_z$. These vectors will be the reference in their coordinate axes.
- Compute the maximum, minimum and center of the frames
  - Repeat for all three frames
- With the obtained statistical data on all three frames, draw the bounding box with max and min values of x, y, z frames. This results in a cuboid
- Apply the transformation and rotations to the cuboid obtained from Eigen vectors and covariance matrix.

The bounding box, as illustrated in Figure 3.12 estimates the normal to the plane in which the Point Cloud data is available in dense. A good grasp is realizable only if the approach vector of the Robot's end-effector is normal to the direction of object orientation. This demonstrates that PCA can estimate the grasp orientation of any

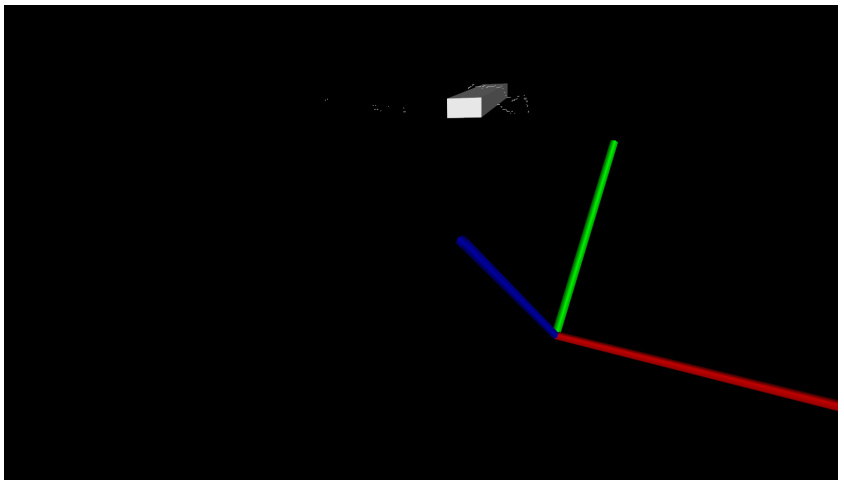


FIGURE 3.12: The grasp approach vector for the handle. The sparse white points represents the handle indices and the cuboid represents the normal to handle surface. The coordinate frames are arbitrary for Eigen values calculation

object from any direction given the Point Cloud indices of the object. Chapter 5 will explain the motion planning approach to demonstrate this task.

Chapter 4

SELF FEEDING TASK

Self feeding is one of the primary ADL tasks that impose a compulsion of using Robot Assistive Technology. As mentioned in Section 1.4, a huge number of people require assistance in feeding themselves. Hence, there is a need to design a system that will autonomously feed people by , given a spoon and a bowl with food. This section describes and illustrates the perception framework developed to realise the self feeding task.

The self feeding task also implements many techniques used in door opening task (Section 3) like downsampling the 3D cloud data, normal estimation and RGB-D Edge detection, octree implementation for faster search, clustering and so on. In order to avoid redundancy, many of the above mentioned steps which had similar procedure are not discussed in this chapter. This chapter contains only specific sections that relates only to self feeding task.

4.1 ROS architecture of the system

This task doesn't need to control the position of the wheelchair and therefore can remove the navigation component from the ROS architecture shown in Figure 3.2. The architecture of the system for self feeding task is shown in the Figure 4.1. The boxes with dashed line represents the group of nodes differentiated by their functionality. The boxes with continuous line represent a node. The arrow mark shows the communication between the nodes through ROS communication pipeline.

The main focus of this section is describing the perception framework developed for robust spoon detection enclosed by a bowl surface. Hence, this chapter covers the functioning of Detection, *Pose Estimation* and *Optimal Grasp Estimation* nodes. *Grasp Execution* and *Motion Planning* nodes are discussed in Chapter 5.

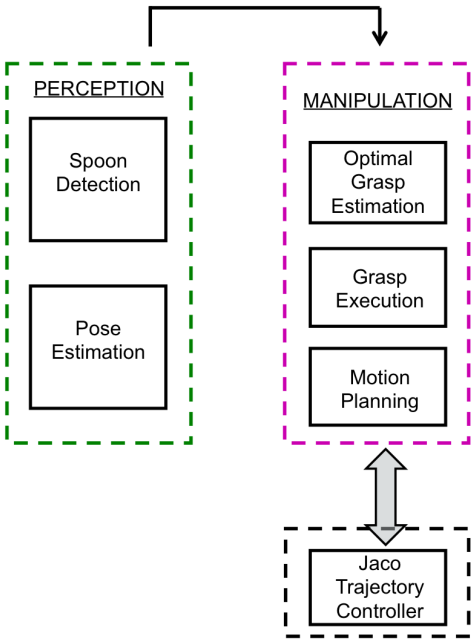


FIGURE 4.1: ROS architecture of self feeding system representing communication between the nodes

4.2 Task Overview

Figure 4.2 shows an example of a workspace with the bowl and spoon in focus. The assumptions of this task is that the color of the spoon and bowl need not contrast each other, but should not be of the same color. Being same color will lose out information while RGB-D processing. Another assumption of this task is that the bowl with spoon is within the reach of manipulator's workspace and in the field of view of primesense device.

4.2.1 Computational Problems

As discussed earlier in chapter 3.3.1, the intensity variations cannot be taken into account because of the varying lighting condition in the home environment and the difference in material with which the spoon is made. As a result, this task should depend only the geometrical data of the scene along with object color obtained from pcl::PointCloud$\langle pcl :: PointXYZRGB \rangle$ data.

This task relies on the 3D geometrical model of the spoon. Spoon can be approximated to a rod in 3-dimensions. There can be few other false positives in the environment as shown in the Figure 4.3. The algorithm should be robust enough to efficiently remove these false positives and extract only the pose of the spoon for grasping.

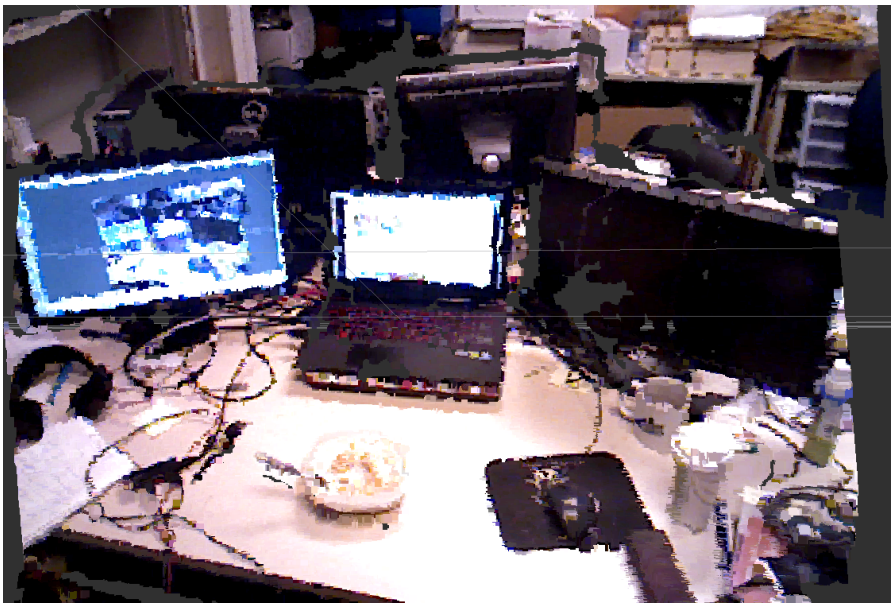


FIGURE 4.2: Workspace of the self feeding task displayed in RViz. The Point Cloud data of bowl and the spoon can be seen

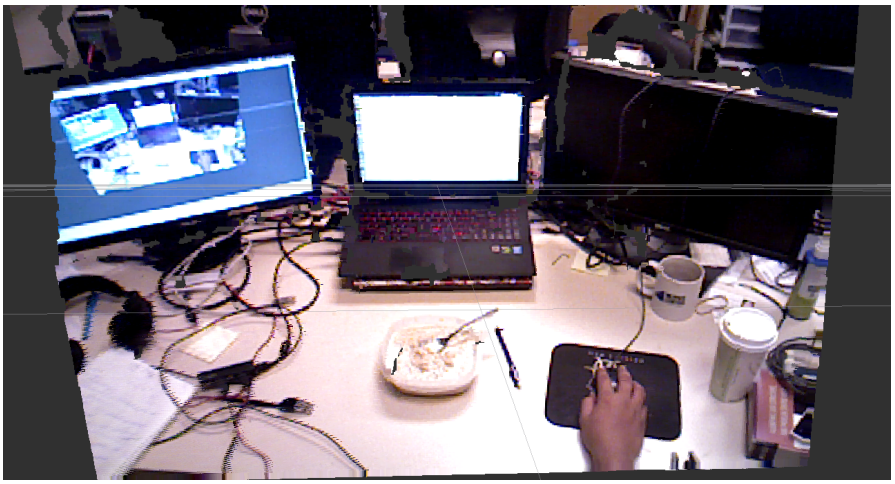


FIGURE 4.3: False positive (a pen) for a spoon like 3D geometry

4.3 Normal estimation and 3D Edge Detection

The approach used for RGB-D edge detection is similar to the approach mentioned in Section 3.4. Refer to that section for a detailed description of surface normal estimation, 3D edge detection using Octree search. Figure 4.4 shows the 3d edge detection results on the workspace containing a spoon inside a bowl. Both the figures 4.4(a) and 4.4(b) illustrate the same scene, but with a change in spoon orientation, just to check if the edge detector is efficient enough to detect the occluded edges. As you can see, the change is detected and the left image shows the entire outline of the spoon although the spoon has been partially immersed in food.

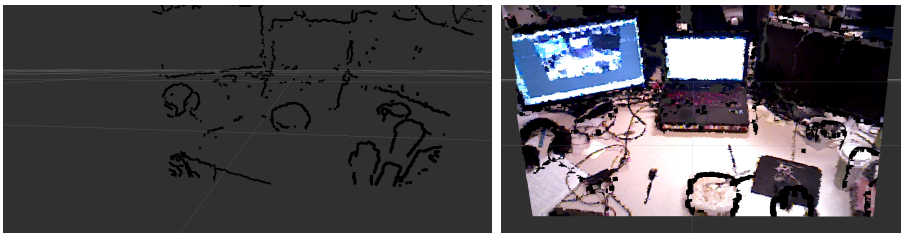


FIGURE 4.4: RGB-D edge detection results Left: 3D indices of extracted Point Cloud edges Right: Edges shown with the environment

The least square fitting estimation problem used for determining normal $\overrightarrow{n}_i = \{n_x, n_y, n_z\}$ to a point $p_i \in \mathcal{P}$ is not explained here as it has been already covered in Section 3.4. Creating a k-d tree for the Point Cloud $\mathcal{P}$ and searching for neighbors $\mathcal{P}_i^k$ for a point p_i is explained in Section 3.4.1.

4.3.1 Removing outliers and false positives

From the obtained indices of 3D edges, extraction of only useful points should be performed. The approach followed to extract the spoon information alone is similar to approach explained in Section 3.4.2 except for some changes in search and model fitting parameters. A spoon in 3-dimensional space can be approximated to a cylinder and RANSAC (see section 3.4.2) will implement a *sample_consensus* search on a Cylindrical model. This would not only extract the spoon indices, but also construct a 3D cylinder over the best fit points of $\mathcal{P}$. The orientation of cylindrical model will give the position and orientation of spoon indices in 3-dimensional space with respect to an arbitrary coordinate frame that can be transformed with respect to camera frame. Thus pose estimation problem of spoon is solved.

But this is not as simple as it seems. There will be many clusters of objects that can be approximated into a 3D cylinder from *sample_consensus*. There can be objects similar to spoon like pen, ointments, wires and so on. An household environment can never be approximated as a noise free environment. As discussed in Section 4.2.1, there can be multiple false positives detected. This approach efficiently removes the false positives by combining the information of the bowl/plate on which the spoon is usually placed. Probabilistic Hough transforms is applied over the edge indices data to extract the vertical and horizontal lines. When these lines put together, the candidate that forms a closed surface, a box, is the indices of the plate.

If the bowl/plate is circle, the indices data can be obtained after applying Hough circles, a parametrized Hough Transforms with the parametric equations

$$x_i = a + R\,cos(\theta)$$

$$y_i = b + R\ sin(\theta)$$

The above equation represents a circle with radius R and center (a, b) , with θ being the angle swept. When θ sweeps to full 360 °, the points (x, y) trace the perimeter of the circle. Finding the parameter triplets (a, b, R) in 3D parameter space is memory and time expensive. The search will be less expensive if any of the parameter, let's say radius R is known as it converts the search problem into 2D parameter space. This approach does not uses 2D parameter search as it is clearly stated that this perception framework needs no prior knowledge.

Once the potential candidate for bowl/plate is estimated, the indices are checked for occlusion or enclosure with the sampling consensus cylindrical model. The cylindrical model which occludes the indices obtained from Hough parametrization will be final candidate for spoon detection and the transformation matrix of the model gives the pose of the spoon. Figure 4.6 shows the extracted candidate from probabilistic Hough Transform in 3D space along with the spoon. More details about the model fitting and grasp selection will be discussed in Section 4.4.

FIGURE 4.5: Extracted indices from parametrized Hough transform displayed in RViz

4.4 Optimal Grasp Selection

After the pose estimation of the object to be manipulated is achieved, estimating the correct approach vector for the robot's end effector to grasp the object should be estimated. The grasp estimation is the last module of the perception framework developed in this thesis. This section explains the approach used for estimating the approach vector for JACO manipulator, with parallel gripper opening, to reliably grasp the spoon.

4.4.1 Definition of optimality

Optimality of a task depends on the resources or time utilized by an agent to complete the task. In this context, an optimal grasp of the spoon is such a grasp which minimizes the overall cost of the solution. Minimizing the cost of a grasping and manipulation problem implies minimizing the number of joint motions a robot has to do to perform a task. More details about the cost function of a manipulation problem involving constraints is discussed in Section 5.1.

Optimal grasping a spoon implies that the manipulator needs to do a minimum joint motions to grab food and reach the user. The optimal way identified to grasp the spoon is the approach vector of the manipulator's end effector being normal to the surface 2D plane fitted along the spoon such that the surface is perpendicular to the ground plane. This implies that the approach vector should always be parallel to the ground surface with the gripper opening-closing direction being perpendicular to the surface of the spoon. Figure 4.6 illustrates optimal grasping position of spoon with JACO. The

FIGURE 4.6: Optimal end-effector grasp pose for self feeding task

reason for choosing this as an optimal grasp pose, because this pose requires just one rotational motion *roll* of the end effector to grab food from the bowl. Also, a very few roll and translation motions will achieve the task of grabbing food into an empty spoon

from bowl. As a result, the perception algorithm developed tries to compute this grasp direction, if feasible, based on spoon's position estimation.

4.4.2 Approach vector

Consider the 3-dimensional coordinate axes to be as shown in the figure 4.7(a). Let the blue colored axis, referred as z-axis, be the axis along the ground/table plane. As discussed in section 4.3.1, a cylindrical model can be fitted onto the spoon by *sample_consensus*. Fitting a 2D plane through the axis of the cylinder and perpendicular to the ground plane, will result in obtaining a plane normal to the ground plane. This 2D plane contains the position and orientation details of the normal of the axis of cylindrical 3D model of spoon. Figure 4.7(b) shows the approach vector represented by the black arrow, which is a vector normal to the center of the 2D plane. The red thin lines shows the JACO's parallel gripper opening and closing direction. This approach vector is maintained parallel to the ground plane if a feasible grasp exist. This grasp approach provided the results (see chapter 6) that proves it to be the optimal grasp pose and can be used for any manipulator just by changing the gripper opening/closing direction accordingly.

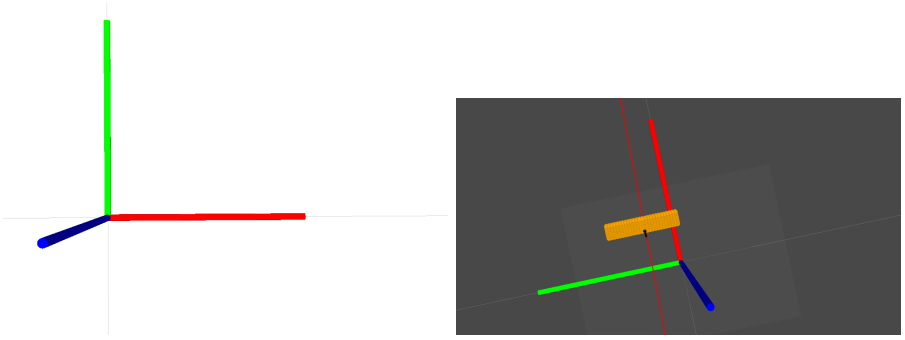


FIGURE 4.7: Grasp estimation Left: The arbitrary coordinate axes with blue axis being the table plane Right: The optimal grasp approach vector is shown by black arrow

Given the Pose of the object and the grasp approach vector, the motion planner will execute the grasp on the object being manipulated. Chapter 5 explains the motion planning approach along with the task planner to execute the manipulation tasks.

Chapter 5

MANIPULATION PLANNING USING TRAJECTORY OPTIMIZER

Trajectory Optimizer, referred as TrajOpt, is an optimization based framework for planning robot motion. TrajOpt was chosen as the motion planner because of its constraints satisfaction capabilities in the task space manipulation. TrajOpt generates the trajectories through sequential convex optimization, a local optimization method for nonconvex problems that requires convex optimization. TrajOpt molds the problem as a sequential convex programming problem and solves it using a convex optimization solver Gurobi [1]. Gradient Optimization Techniques for Efficient Motion Planning(CHOMP) [20] and Stochastic trajectory optimization for motion planning (STOMP) [21] are other two optimization based planning framework that can generate high quality path from an initial seed trajectory. The limitations with TrajOpt and other optimization based planners are that they may get stuck in local optima. This problem can be addressed by running more iterations for optimization solver or providing a good and dynamically feasible initial seeds.

5.1 The Optimization Problem

The tasks are executed by JACO manipulator (see Appendix A for details), a 6 Degree of Freedom robot. The desired Cartesian space motion of the manipulator is specified in terms of a 6x1 pose vector.

[1] http://www.gurobi.com/products/gurobi-optimizer

$$Pose\ vector = [x \quad y \quad z \quad pitch \quad roll \quad yaw]^T$$

The first three entries represents the end effector position in the 3-dimensional Cartesian space $\{x, y, z\}$ while the last three entries represents the orientation of the end effector. The optimizer formulates these Cartesian motion as its cost and constraints, also accepting additional constraints, and computes a trajectory represented by a set of time-stamped waypoints. The non-convex optimization problem can be formulated as

$$\min_x f(x)$$

$$s.t \;\; g_i(x) \geq 0, \; i = 1, 2, ...m$$

$$h_j(x) = 0, \; j = 1, 2, ...p$$

with the optimization variable $x \in \mathcal{R}^n$, n being the degree of freedom of the manipulator. If the trajectory has $\mathcal{T}$ waypoints, then x can be represented of the form $x = q_{1:T}$, where $q_t \in \mathcal{R}^n$ describes the joint configuration at the $t-th$ timestep. f and g_i are non-convex scalar function and h_j is non-affine. The commonly used inequality constraints in motion planning problem are collision avoidance, joint limits constraints and speed limits in Cartesian space. Common equality constraints include the end effector pose and orientation constraints which will be discussed in section

The cost function is represented in configuration space and it is written as:

$$f(q_{1:T}) = \sum_{t=1}^{T}((q_{t+1} - q_t)^T Q_1 (q_{t+1} - q_t) + (q_t - q_{nom})^T Q_2 (q_t - q_{nom})$$

where $Q1, \;\; Q2 \;\; \geq 0$ and q_{nom} contains the configuration of nominal posture. The above equation enforces cost penalty on joint-space velocities as it tries to penalize a high velocities by maintaining a difference check between q_{t+1} and q_t. In practice, the collision check is included in the cost function rather than a constraint satisfaction. This is because, the robot's end-effector must come in contact with the object being manipulated, and if the collision check was a constraint, this action would never be validated. If the collision check is included as a cost parameter, the penalties can be handled off for the collision with the object being manipulated.

5.2 Motion Planning and execution

Motion planning for a specific task like door opening and self feeding can be achieved through a high-level task planning approach that determines long term strategies like picking up a spoon from anywhere outside the bowl, manipulating it inside the bowl and grabbing food. The repetitive execution of motions from bowl to user for self feeding can be performed by a low-level motion planner. Combining the low-level motion planner and high-level task planner [22] is a hard problem because of the generality of the task planner. Implementing a task planner is out of the scope of this thesis and assumptions are made that the preconditions are satisfied to execute motion planning. The motion planning approach for door opening task and self feeding task is discussed in Section 5.2.1 and 5.2.2 respectively.

5.2.1 Door Opening Task

Chapter 3 described the approach to detect the 6 dimensional pose of the handle and computed the approach vector based on the 3D orientation of the handle using Principle Component analysis. Given a pose and the approach vector, the motion planner computes a set of waypoints $q_{1:T}$, where q_t is the joint configuration of the robot at time $t \in \{1, T\}$ to execute the motion of grasping the handle and turning it to open the door.

The Planning Domain Definition Language (PDDL) [23] helps to formulate the problem by defining a deterministic task planning problem as a tuple $\langle \mathcal{A}, s_0, g \rangle$ where $\mathcal{A}$ is a set of parameterized action, s_0 is the initial state and g is the goal state. The state transition function f specifies the transition from the current state x to the next state x' when an action $u \in U(x)$, the action space, is applied. The task of a planning problem is to find a finite sequence of robot actions, that when applied to the robot trasforms the initial state s_0 to the goal state g. Every action, when given a present state, applies a state transformation given by the effect of the action. For example, a discrete *grasp* action on handle using current gripper state could be represented as follows :

```
grasp(handle, gripper)
    precon  Empty(gripper)
    effect  InGripper(handle), ~Empty(gripper)
```

All other robot discrete action like *turnHandle, leaveHandle*results with an effect on the states and can be represented in a similar way.

A motion planning problem is represented as a tuple $\langle \mathcal{Q}, f, p_0, p_t \rangle$, where $\mathcal{Q}$ is the configuration space containing all the possible configuration of the robot, f is a boolean

function that determines whether a current configuration $q_t \in \mathcal{Q}$ is in collision and p_0, p_t is the initial and final poses. A collision-free motion plan solving a motion planning problem is a trajectory in $\mathcal{Q}$ from p_0 to p_t such that f doesnt hold for any pose in the trajectory. The collision with the handle is allowed by modifying f to be $false$.

The door opening manipulation task can be divided into three sub-tasks :

- Grasp the handle
- Turn the handle to unlatch
- Pull the handle to open the door

The last sub-task to pull the handle and open the door is not performed in this task as JACO's gripper possesses fragile fingers which is designed to perform light load task. Pulling the heavy door with those fingers is not feasible and hence was not tried. The task can be represented as discrete actions as follows:

```
grasp(handle, gripper, pose1, pose2, traj1)
    precon  Empty(gripper), At(gripper, pose1)
            isGraspPossible(pose2, handle), setConstraints(traj1)
            isMotionPossible(traj1, pose1, pose2)
    effect  In(handle, gripper), ~Empty(gripper)
            At(gripper, pose2)

turn(handle, gripper, pose2, pose3, traj2)
    precon  In(handle, gripper), ~Empty(gripper)
            At(gripper, pose2), setConstraints(traj2)
            isMotionPossible(traj2, pose2, pose3)
    effect  In(handle, gripper), ~Empty(gripper)
            At(gripper, pose3)
```

The structure of discrete action is similar to [22], where they try to pick an object from one position and place it at another position. I had introduced few predicates like $isGraspPossible$ and $isMotionPossible$ as precondition checks which returns boolean true or false. $isGraspPossible(pose_2, handle)$ returns true if $pose_2$ is a pose at which $handle$ can be grasped. $isMotionPossible(traj_1, pose_1, pose_2)$ returns true only if motion planning action tuple $\langle \mathcal{Q}, f, pose_1, pose_2 \rangle$ holds for the trajectory $traj_1$. That is, $traj_1$ should give feasible solution from $pose_1$ to $pose_2$. $In(handle, gripper)$ returns true if the gripper is currently grasping the handle. $At(gripper, pose_2)$ holds true of the end effector is in $pose_2$. Remaining predicates used are self-explanatory.

End effector poses used above are 6x1 vectors. $pose_1$ is the home configuration of the robot. $pose_2$ is the configuration of the robot that grasps the handle. $pose_3$ is the configuration of the robot after turning the handle to unlatch the door. The end effector

pose for different poses are shown in the Figure 5.1. The rotation of handle is achieved by setting the end effector pose constraints while TrajOpt calculates the trajectory. The end effector pose constraints are set by the by calculating the corresponding quaternion, a 4x1 vector of $\langle w, x, y, z \rangle$. An array of position and quaternion of end-effector constitute a set of waypoints and that gives the trajectory for the robot to execute.

FIGURE 5.1: End effector poses during the task execution Left:Home Pose (Pose1) Middle: Pose for grasping handle (Pose2) Right: Pose after turning handle (Pose3)

5.2.2 Self feeding task

Chapter 4 described the approach to detect the 6 dimensional pose of the spoon and computed the approach vector based on the 3D orientation of the spoon. Given a pose and the approach vector, the motion planner computes a set of waypoints $q_{1:T}$ to execute the motion of grasping the spoon, grabbing food from the bowl, feed the user and repeat the task.

The robot motion is executed through a sequence of several feasible states. Discrete planning (discusses in section 5.2.1) formulates the planning problem for the self feeding task. The problem formulation and discrete actions are similar to what section 5.2.1 explains. They are not discussed in this section to avoid redundancy.

The self feeding manipulation task can be divided into three sub-tasks :

- Grasp the spoon
- Translate and rotate the spoon to grab food from bowl
- Manipulate the spoon to the user and execute feed
- Manipulate back to initial grasp position (and repeat steps 2 - 4)

These sub-tasks can be represented as discrete actions as follows :

```
grasp(spoon, gripper, pose1, pose2, traj1)
    precon  Empty(gripper), At(gripper, pose1)
            isGraspPossible(pose2, spoon), setConstraints(traj1)
            isMotionPossible(traj1, pose1, pose2)
    effect  In(spoon, gripper), ~Empty(gripper)
            At(gripper, pose2)

grabFood(spoon, gripper, pose2, pose3, traj2)
    precon  In(handle, gripper), ~Empty(gripper)
            At(gripper, pose2), setConstraints(traj2)
            isMotionPossible(traj2, pose2, pose3)
    effect  In(spoon, gripper), ~Empty(gripper)
            At(gripper, pose3)

feedUser(spoon, gripper, pose3, pose4, traj3)
    precon  In(spoon, gripper), ~Empty(gripper)
            At(gripper, pose3), setConstraints(traj3)
            isMotionPossible(traj3, pose3, pose4)
    effect  In(spoon, gripper), ~Empty(gripper)
            At(gripper, pose4)

traceBack (spoon, gripper, pose4, pose2, traj4)       %Repeat - back to grab food
    precon  In(spoon, gripper), ~Empty(gripper)
            At(gripper, pose4), setConstraints(traj4)
            isMotionPossible(traj4, pose4, pose1)
    effect  In(spoon, gripper), ~Empty(gripper)
            At(gripper, pose1)
```

All the assumptions and task predicates defined are similar to definition used in Section 5.2.1. Each sub-task effects in manipulation of robot from one pose to another pose of end-effector. The end effector pose for different poses are shown in the Figure 5.2. $pose_1$ is the home configuration of the robot. $pose_2$ is the configuration of the robot that grasps the spoon. $pose_3$ is achieved from $pose_2$ after executing translation (if needed) and rotation to grab the food. $pose_4$ is the configuration of the robot that feeds the user. The robot takes a pause of 10 seconds to ensure that the user has reached to spoon to eat. And these set of sub-tasks are repeated to feed the user. The costs and constraints are set similarly (not the same) as explained in Section 5.2.1. Self feeding task has a hard constraint of not spilling the food while manipulating from bowl to user. $traj_3$ is responsible for this manipulation. Rotation of the end effector is only allowed along the axis perpendicular to the ground plane as this rotation will not spill the food. The remaining two rotations *roll* and *pitch* are not allowed. This can be done by adding a rotational constraint to the end effector along the entire trajectory. The rotation coefficient of the trajectory, rot_{coeffs} are set to [k,0,0], where k is a very small constant allowing minimal rotation along x-axis (ground being z axis). The trajectory $traj_3$ accepts no difference in rotational component *roll* and *pitch* between $pose_3$ and

FIGURE 5.2: End effector poses during the task execution Left:Home Pose (Pose1) Middle: Pose for grasping spoon (Pose2) Right: Pose to feed the user (Pose4)

$pose_4$. Even if a minimal difference exists, TrajOpt will either return a trajectory with infinite cost or will get stuck in a local minima that never ends the search problem.

5.3 Pipeline of the overall system

Planning the robot motion and execution completes the task of opening the door/self feeding successfully. The entire perception and manipulation planning framework is built on ROS for modularity and easy communication. The entire ROS pipeline to complete the task is shown in Figure 5.3.

A ROS node is an executable that communicate with other nodes using ROS topics. In the above figure, Primesense and JACO manipulation are sensor and actuator respectively. Other solid blocks represent different ROS nodes. The algorithms developed for this thesis can be split into different ROS nodes as shown for modularity reasons. The nodes enclosed in red boxes are related to perception algorithms and the nodes enclosed in green boxes are for manipulation capabilities.

The content in between the boxes (shown above arrows) are the type of ROS messages being communicated. For example, *geometry_msgs/Pose* is a ROS message type that contains both position (*geometry_msgs/Point*) and orientation (*geometry_msgs/Quaternion*) information.

The Primesense sensor publishes (sends) the Point Cloud data of type *sensor_msgs/PointCloud2* that is used by the Handle detector node to process and detect the handle. The dashed

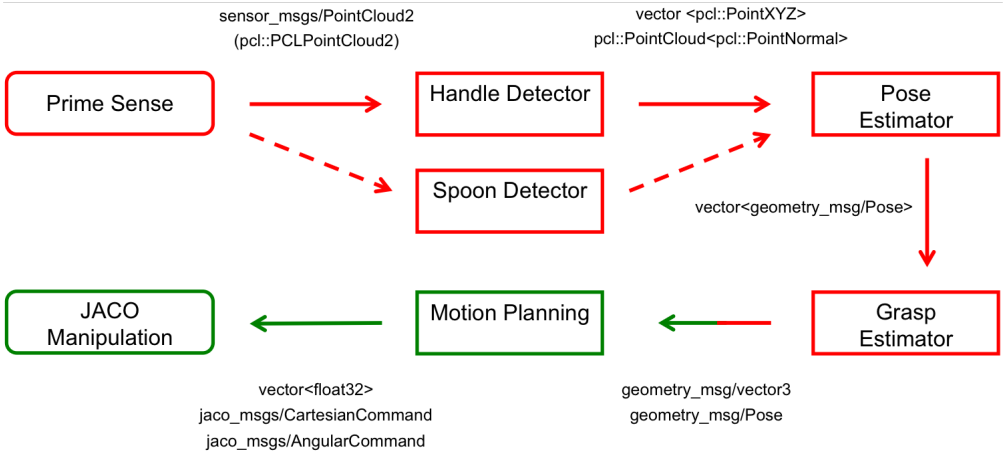


FIGURE 5.3: ROS pipeline of the entire system showing communication between nodes (in boxes). The content in between transition shows the ROS message type being published

arrows represent an alternate task of self feeding. The handle detector node computes the indices that contains the handle points as *vector ¡pcl::PointXYZ¿* along with *pcl::PointNormal* information to estimate the orientation of detected objects. The Pose estimator computes an array of possible Poses of the object in the world. The Grasp estimator node compute the approach vector as a *geometry_msgs/vector3* message type. With the Pose and approach vector information, the motion planning node computes an array of commands for the JACO arm to manipulate and complete the manipulation task.

Chapter 6

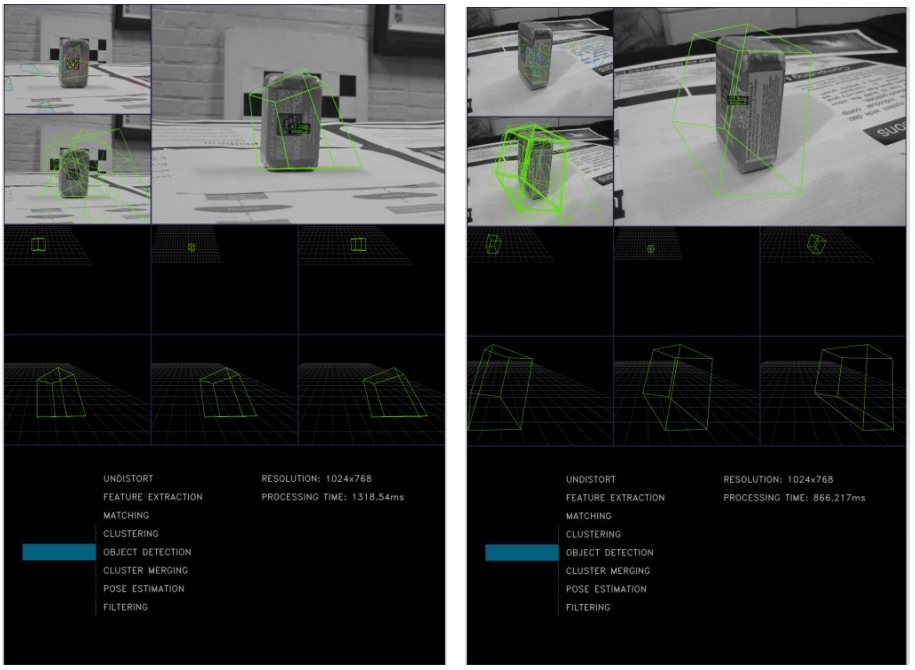
EXPERIMENTS AND RESULTS

This chapter presents the recorded experimental data to evaluate the algorithms and methods discussed in the earlier chapters. The data provided in the following sections are mostly related to success/failure rates over numerous runs. The results of MOPED framework, the door opening task and self feeding task is discussed.

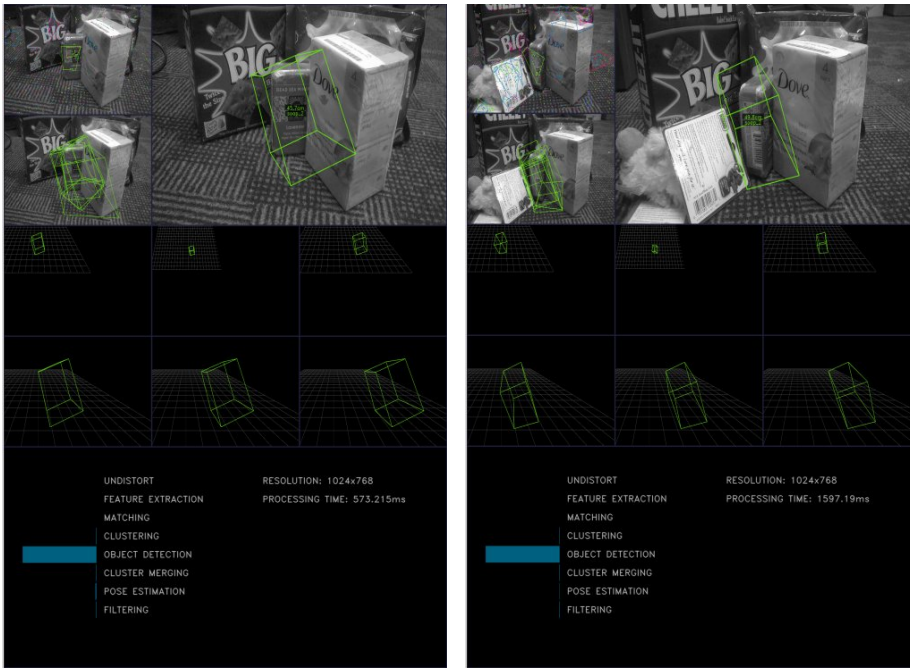
6.1 Multiple Object Pose Estimation and Detection results

In the tests that were performed on Pose Estimation of the soap bar, I was successfully able to detect the soap bar and estimates its 6D pose in 72 images out of 75. These images had set of images from different cameras, locations, lighting conditions, cluttered and uncluttered environments. Images a) and b) in Figure 6.1 displays detection of the soap when only soap is present in the scene and the images c) and d) displays detection when soap is partially occluded by other objects.

Given the 6D pose of the object in the scene, the grasp pose is estimated and manipulator could successfully manipulate the object. This 6D pose estimation technique is highly reliable.



(A) Soap Front (B) Soap Side



(C) Cluttered Environment Front (D) Cluttered Environment Side

FIGURE 6.1: Detection of soap bar using MOPED

6.2 Door opening results

The door opening task has several sub-tasks like handle detection, pose estimation, grasp estimation and motion execution tasks performance was evaluated. Given the point cloud data, the handle can be efficiently located and opened. Figures from section 3.4, section 3.5 and section 5.2.1 illustrates the results of these sub tasks being executed successfully. The handle can be detected and opened from multiple angles lying within a region where a good view of the handle can be obtained. Principle Component Analysis finds the Eigen Vector of the extracted cloud indices to estimate the grasp approach vector.

TABLE 6.1: Number of successful attempts to execute the sub-tasks of door opening task. Number of attempts = 10

Sub-tasks	Successful attempts	Success %
estimate handle pose	9	90%
grasp handle	7	77.8% (7/9)
turn handle	7	100% (7/7)

The table 6.1 shows the successful attempts the robot executed the sub-tasks without any time constraints. The overall task success rate = 70 % (7/10 attempts).The handle detection and its pose estimation was a very successful task and that proves the reliability of the algorithm developed. Grasp estimation failed in few cases where the organization of extracted handle point indices are disoriented with the reference frame. This is not likely to happen in indoor navigation because the user on the wheelchair will approach the door in a direction normal to the door plane by allowing some considerable deviation. The handle grasping task failed in attempts when the point cloud data does not return the accurate Pose $\mathcal{P}$ and grasp approach vector. This happened due to the noise in the extracted point data due to presence of other handle-like features in the scene. Once the handle grasp is achieved, turning the handle is achieved always as TrajOpt generated accurate trajectories to turn the handle.

Figure 6.2 shows the handle detection algorithm being tested under various environmental conditions like varying light intensity, a different type of handle (knob) and varying field of acquisition. The results were obtained for these variation to show that the algorithm is robust to be implemented in household environment.

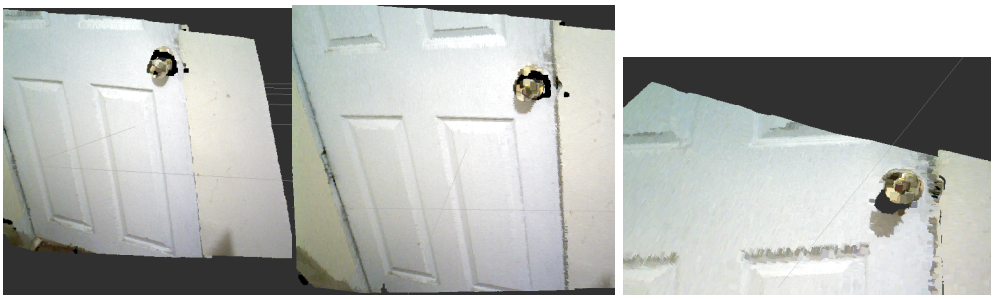


FIGURE 6.2: Handle detection algorithm tested under various conditions

6.3 Self feeding results

The self feeding task was executed in varying conditions like different spoon poses, different number of false positives, different bowl and spoon models and so on. As the algorithm developed requires no prior model of any object, the task could be executed with different model of spoon and bowl. However in some cases, the grasp approach vector could not be efficiently obtained due to the cylindrical nature of the spoon. In such case of ambiguity, the grasp approach vector is manually hard-coded to be the vector parallel to the ground plane as it is the optimal approach direction for self feeding task (see Section 4.4). In such cases, the algorithm loses its robustness. The table 6.2 shows the success ratio when the self feeding task was performed under varying conditions.

TABLE 6.2: Success ratio for self feeding task. Number of attempts = 10

Sub-tasks	Successful attempts	Success %
estimate spoon pose	7	70%
grasp spoon	6	84.4% (6/7)
grab food	6	100% (6/6)
feed user	5	83.3% (5/6)

The overall success ratio for the task is 50% (5/10 attempts). This low success rate is because of the thin geometry of spoon in large point cloud. Also, most of the household object which could potentially be in the scene can also be fitted into a 3D cylindrical model. This results in large number of outliers and false positives. However, this can be improved if the spoon model is learned by the algorithm, which is against the assumptions made for this algorithm.

Chapter 7

CONTRIBUTION AND DISCUSSION

The manipulation planning for JACO (Chapter 5) used in validating the perception algorithms (Chapter 3 and 4) was developed by Xianchao Long from Robotics and Intelligent Vehicles Research (RIVeR) laboratory. He developed the motion planning framework for the above discussed ADL manipulation tasks. The MOPED framework (Chapter 2) was developed for a course project and the approach was extended to obtain a feasible grasp pose for JACO manipulator from the obtained 6D pose.

The approach presented in this thesis demonstrates to perceive unknown and known objects. As of now, this work doesn't guarantee a complete autonomy for dealing with unknown objects as the algorithm is not sophisticated to perceive and classify unknown objects. Instead, the perception algorithms developed are task specific and can classify and estimate the pose of the objects that are related to the task. As discussed in the motivation of this research, this algorithms are developed under the classification of Human in the Loop Cyber Physical System (HiLCPS). The Brain Body Control Interface (BBCI), researched and developed in Robotics and Intelligent Vehicles Research (RIVeR) laboratory serves as the cyber medium. The user should think to open the door run the door-opening task related algorithms and the pose estimator will only look for handle like features in a door plane ignoring all other geometry. That is, self feeding task and door opening task are completely independent to each other.

The door opening task for indoor navigation provided similar results like the Laser based door opening with a PR2 [24] from Laser and intensity data , but the algorithm demonstrated in this thesis did not rely on the intensity data of the Point cloud. This algorithm will work in almost all household conditions allowing a cushion of few false

positives. The motion planner, TrajOpt, validates the task successfully by taking all constraints required for a household manipulation task.

Door traversal task using ATLAS Humanoid robot [25], by team WPI-CMU, was one of the tasks at DRC finals. ATLAS successfully identified the door, walked towards the door, detected handle, turned the handle, opened the door and walked past the door. The handle detection was done by color segmentation based on the assumption that the handle is of significantly different color from the door. But the algorithm developed for handle detection in this thesis does not rely on color or intensity changes in order to make it robust for varying environmental conditions at home.

To the best of my knowledge, there has been no published successful work on autonomous self feeding using robots. The algorithms developed as a part of this thesis towards feeding the user will stand as a benchmark for future work on similar goals. The results of this thesis can be reproduced in simulation using ROS and openRAVE simulator. The future work on this task is to develop a highly robust and reliable grasping technique by using advanced machine learning classifiers to detect and grasp the spoon/fork to achieve 100% success rate.

The Point Cloud data were used only for the object classification and pose estimation problems. As the robot manipulates in an environment, point cloud data can be used in developing a probabilistic model of the environment's occupancy data for collision avoidance by creating an Octomap representation. Another future scope of this project is to develop a closed-loop feedback control system by visual servoing. The real time perception data can provide the pose of the manipulator's end-effector and can be used as a feedback for error propagation and correction as it executes its planned trajectory.

Appendix A

JACO arm manipulation capabilities

JACO robotic arm is a 6 -DoF manipulator shown in Figure A.1. Advanced motion planners like TrajOpt can be implemented on the JACO (see Chapter 5). This appendix is written for the readers not familiar about executing a trajectory using a manipulator.

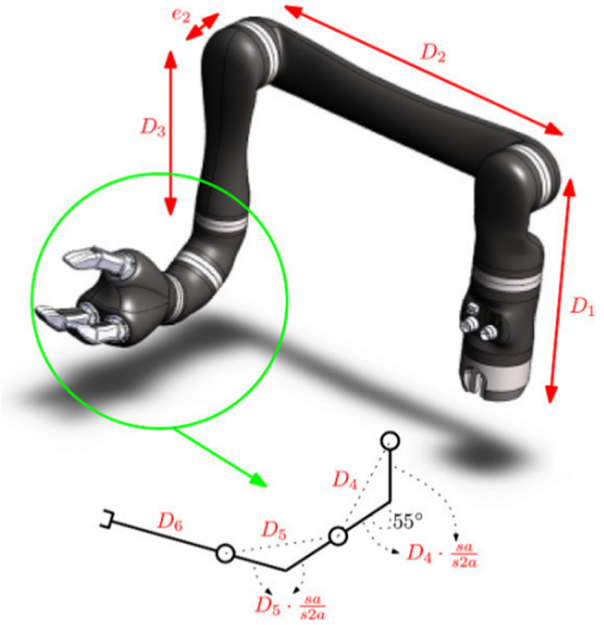


FIGURE A.1: A view of JACO showing the link parameters

A time-stamped trajectory contains the information of position, velocity and acceleration of each joint during every step of the trajectory. Given the position and velocity details of start position and goal pose [1], Trajectory generation can be done by computing a

[1] A pose of the robot represents the position and orientation of the end effector as a 6x1 vector

quintic time dependent polynomial. To represent mathematically, Given the start and end positions $(q_0(t), q_f(t))$, velocities $(\dot{q}_0(t), \dot{q}_f(t))$, and accelerations $(\ddot{q}_0(t), \ddot{q}_f(t))$ for start time, t_0 and end time t_f, the position can be given by the quintic polynomial as follows:

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{A.1}$$

Therefore, velocity and acceleration are given by:

$$\dot{q}(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4 \tag{A.2}$$

$$\ddot{q}(t) = 4a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3 \tag{A.3}$$

We can solve for the coefficients, by plugging in the known start and end positions, velocities and accelerations given by the following linear system:

$$\vec{q}(t) = A \times \vec{a}$$

Therefore,

$$\vec{a} = A^{-1} \times \vec{q}(t) \tag{A.4}$$

which is equivalent to:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} 1 & t & t^2 & t^3 & t^4 & t^5 \\ 1 & t & t^2 & t^3 & t^4 & t^5 \\ 0 & 1 & 2t & 3t^2 & 4t^3 & 5t^4 \\ 0 & 1 & 2t & 3t^2 & 4t^3 & 5t^4 \\ 0 & 0 & 2 & 6t & 12t^2 & 20t^3 \\ 0 & 0 & 2 & 6t & 12t^2 & 20t^3 \end{bmatrix}^{-1} \times \begin{bmatrix} q_0(t) \\ q_f(t) \\ \dot{q}_0(t) \\ \dot{q}_f(t) \\ \ddot{q}_0(t) \\ \ddot{q}_f(t) \end{bmatrix}$$

The equation A.4 provides the values for the coefficients of the quintic polynomial that can then be used for the desired trajectory generation. This is done by creating an array with the range $[t_0, t_f]$ with a step size of 0.1 seconds.The values in the array are then applied to equations A.1, A.2, and A.3 to get corresponding discretized position, velocity and acceleration trajectory profiles. For the JACO arm, the discretized positon trajectory profile was used to control the desired set points of the robotic arm.

The 6 dimensional end effector pose of the above resulted discredited position trajectory profile was recorded and plotted in MATLAB illustrated in Figure A.2. The trajectory obtained results in a baseball swing using a single arm with required final velocity and acceleration required to hit the ball hard. The swing was performed without the baseball stick just for demonstration purpose.

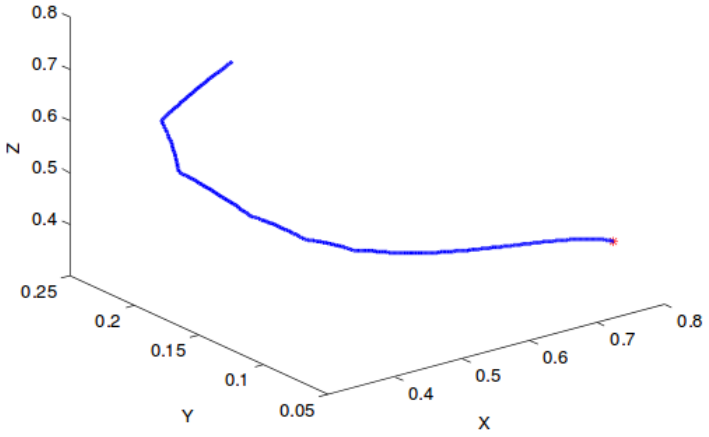


FIGURE A.2: Trajectory of the JACO manipulator playing a baseball swing

Appendix B

Camera Calibration

Camera calibration is the process of finding the true parameters of the camera that took the photographs. The parameters of a camera can be classified as extrinsic and intrinsic parameters. Extrinsic parameters describes the camera's orientation and location is real world. Intrinsic parameters define the relationship between pixel coordinates and camera coordinates. Some of these parameters are focal length, format size, principal point, and lens distortion. Camera Calibration is primarily used when modeling scenes virtually from real inputs. One of the main purpose of camera calibration is to figure out where the camera was in relation to a scene in a photograph. Thus camera calibration is mandatory in 3D computer vision.

In order to achieve the intrinsic parameters, calibration is done using a 2D planar pattern. A chessboard, shown in Figure B.1 was observed by the camera at different orientations. The primary requirement is that the chess board should not be a square, number of rows and columns should be different. This requirement would account for the rotations made by the camera while observing this 2D pattern from various orientations.

'Distortion' is primarily accounted in camera calibration which arises due to two factors namely radial and tangential factors. The distortion due to radial factors can be corrected as

$$\mathrm{x}_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$\mathrm{y}_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

The distortion due to radial factors can be corrected as

$$\mathrm{x}_{corrected} = x + [2p_1 xy + p_2(r^2 + 2x^2)]$$
$$\mathrm{y}_{corrected} = y + [2p_1 xy + p_2(r^2 + 2y^2)]$$

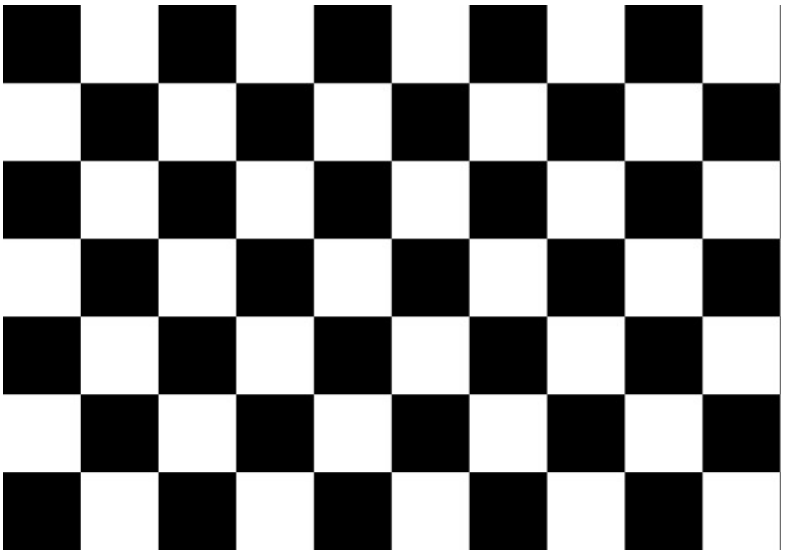


FIGURE B.1: A rectangular chess board used for camera calibration

The position on the corrected output image will have the coordinates $(x_{corrected}, y_{corrected})$. The five distortion parameters can be implemented as 1x5 matrix in OpenCV as

$$Distortion_{coefficient} = [k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3]$$

The 3D point is converted into a 2D projection in the image plane using the matrix

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

The 3x3 matrix that relates the real world coordinates (X,Y,Z) with the image plane coordinates (x,y) is called the camera intrinsic matrix. The unknown parameters f_x, f_y are the camera focal length and c_x, c_y are the optical centers of the image plane expressed in pixel coordinates.

The process of determining the distortion matrix and the camera intrinsic matrix is calibration. The input images are the images of the planar pattern from different orientations. The number of images required depends on the type of the pattern. In case of chessboard, the minimum number of snapshots required is four, but to eliminate the noise in the image, I used seven images for calibration. This gave a better result and proper calibration was done.

Bibliography

[1] G. Schirner, D. Erdogmus, K. Chowdhury, and T. Padir. The future of human-in-the-loop cyber-physical systems. *Computer*, 46(1):36–45, Jan 2013. ISSN 0018-9162. doi: 10.1109/MC.2013.31.

[2] John Schulman, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013. doi: 10.15607/RSS.2013.IX.031.

[3] Juan M. Corchado, Javier Bajo, and Ajith Abraham. Gerami: Improving healthcare delivery in geriatric residences. *IEEE Intelligent Systems*, 23:19–25, 2008.

[4] Paolo Sernani, Andrea Claudi, Luca Palazzo, Gianluca Dolcini, and Aldo Franco Dragoni. Home care expert systems for ambient assisted living: A multi-agent approach.

[5] T. Mukai, S. Hirano, H. Nakashima, Y. Kato, Y. Sakaida, S. Guo, and S. Hosoe. Development of a nursing-care assistant robot riba that can lift a human in its arms. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 5996–6001, Oct 2010. doi: 10.1109/IROS.2010.5651735.

[6] Chih-Hung King and Chen. Dusty: an assistive mobile manipulator that retrieves dropped objects for people with motor impairments. March 2012.

[7] Dmitry Sinyukov, Ross Desmond, Matthew Dickerman, James Fleming, Jerome Schaufeld, and Taskin Padir. Multi-modal control framework for a semi-autonomous wheelchair using modular sensor designs. *Intelligent Service Robotics*, 7(3):145 – 155, July 2014.

[8] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4, May 2011. doi: 10.1109/ICRA.2011.5980567.

[9] Baltes MM and LANG FR. Everyday functioning and successful aging: the impact of resources. *Psychology and Aging*, 12(3):433–443, 1997.

[10] Alvaro Collet Romea, Manuel Martinez Torres, and Siddhartha Srinivasa. The moped framework: Object recognition and pose estimation for manipulation. *International Journal of Robotics Research*, 30(10):1284 – 1306, September 2011.

[11] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. *ACM transactions on graphics (TOG)*, 25(3):835–846, 2006.

[12] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[13] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

[14] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, 2002.

[15] Nioy J Mitra and An Nguyen. Estimating surface normals in noisy point cloud data.

[16] C. Choi, A. J. B. Trevor, and H. I. Christensen. Rgb-d edge detection and edge-based registration. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1568–1575, Nov 2013. doi: 10.1109/IROS.2013.6696558.

[17] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, Nov 1986. ISSN 0162-8828. doi: 10.1109/TPAMI.1986.4767851.

[18] Martin A Fischer and Robert C. Bolles. andom sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381 – 395, 1981.

[19] Jon Shlens. A tutorial on principle component analysis: Derivation, discussion and singular value decomposition. 25 March 2003.

[20] Nathan Ratliff, Matthew Zucker, J. Andrew (Drew) Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, May 2009.

[21] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4569–4574, May 2011. doi: 10.1109/ICRA.2011.5980280.

[22] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 639–646, May 2014. doi: 10.1109/ICRA.2014.6906922.

[23] M. Fox and D. Long. Pddl2.1: an extension to pddl for expressing temporal planning domains. *JAIR*, 20(1):61 – 124, 2003.

[24] R. B. Rusu, W. Meeussen, S. Chitta, and M. Beetz. Laser-based perception for door and handle identification. In *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pages 1–8, June 2009.

[25] Nandan Banerjee, Xianchao Long, Ruixiang Du, Felipe Polido, Siyuan Feng, Christopher G. Atkeson, Michael A. Gennert, and Taskin Padir. Human-supervised control of the ATLAS humanoid robot for traversing doors. In *15th IEEE-RAS International Conference on Humanoid Robots, Humanoids 2015, Seoul, South Korea, November 3-5, 2015*, pages 722–729, 2015. doi: 10.1109/HUMANOIDS.2015.7363442. URL `http://dx.doi.org/10.1109/HUMANOIDS.2015.7363442`.